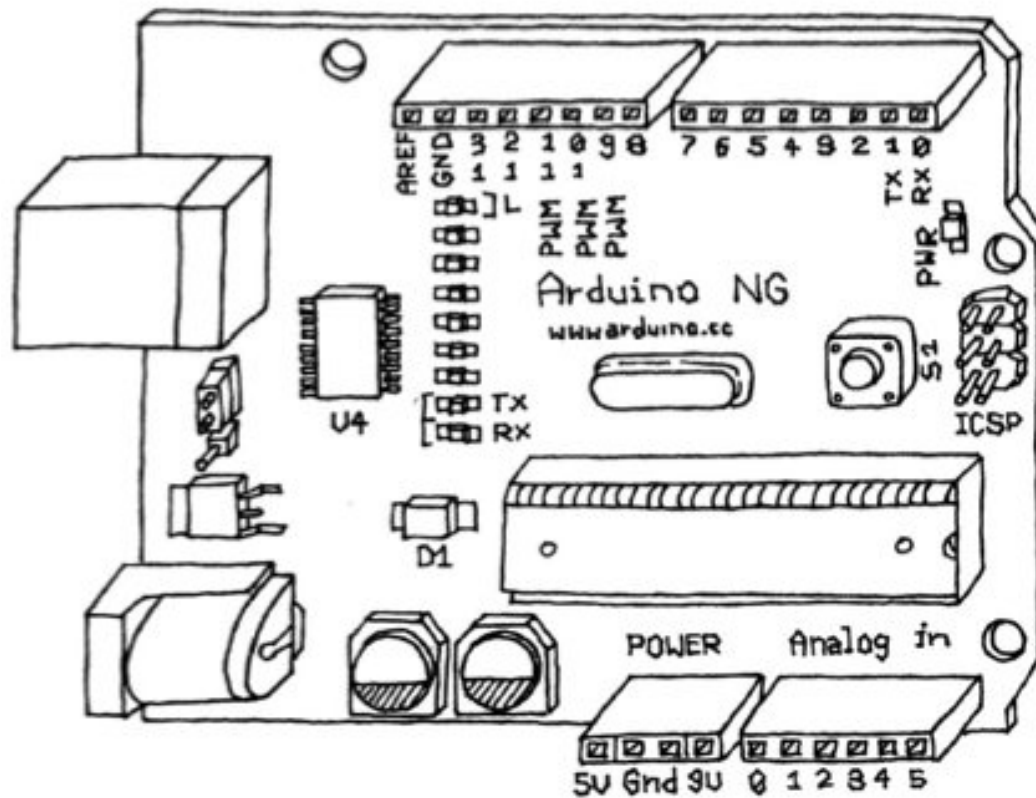


Arduino part A

slides rel. 4.3 - free documentation - 2015



Contents

Part A

- What is Arduino?
- What is an "embedded system"?
- The microcontroller Atmega168 / 328
- The Arduino board
- The software environment
- Applications in C++ and exercises

Part B

- AVR architecture Atmega328

Part C

- ASM

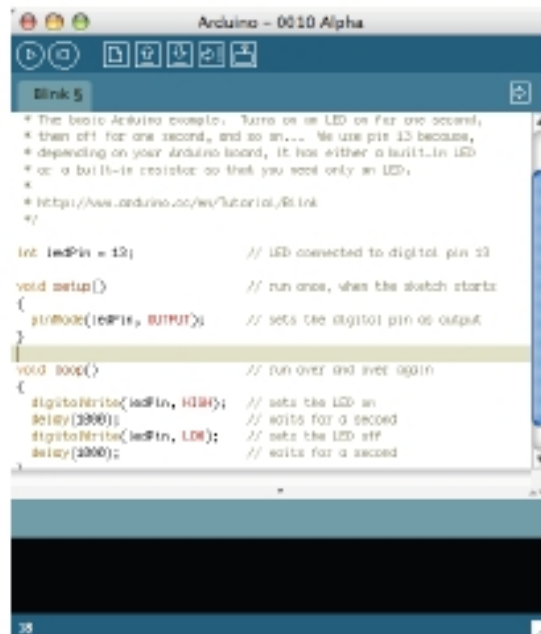
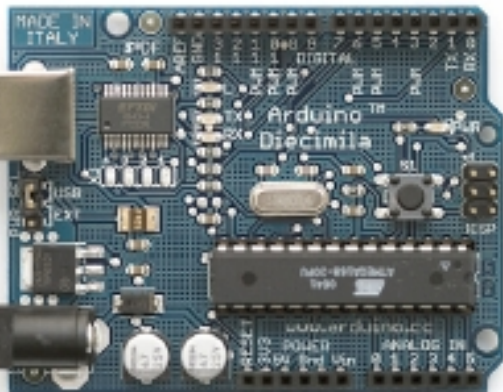
What is Arduino?

The word “Arduino” can mean 3 things

A physical piece
of hardware

A programming
environment

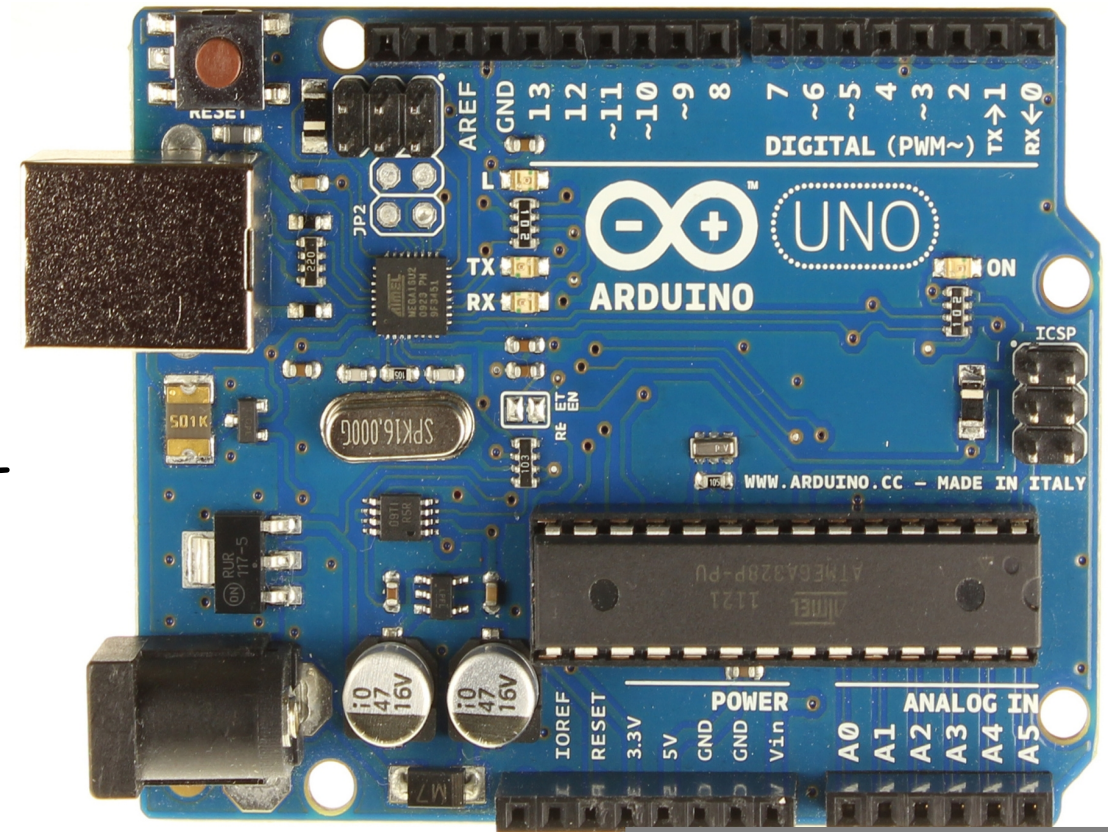
A community
& philosophy



What is Arduino?

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software.

It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments



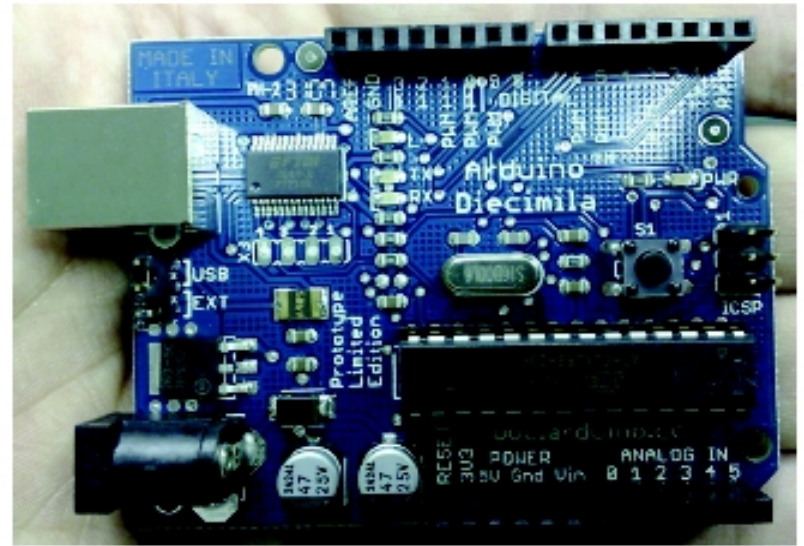
What is Arduino?

- Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators
- The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing).
- Arduino projects can be **stand-alone** or they can communicate with software on running on a computer (e.g. Flash, Processing, ...).

Arduino Capabilities



=

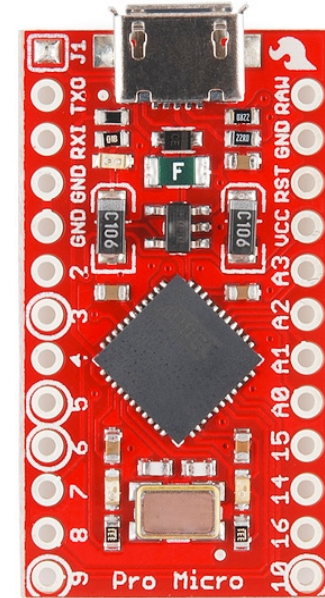
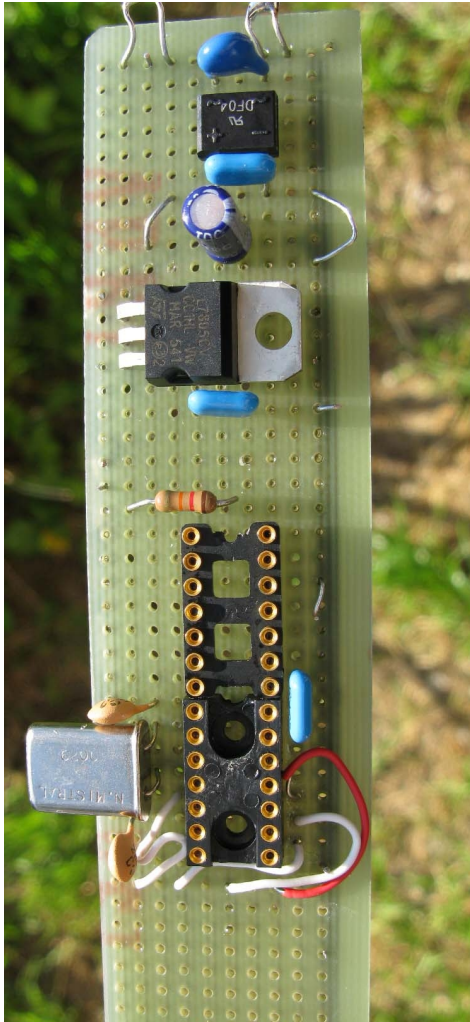


Intel 286

Arduino

PC AT-IBM (1984)

stand-alone boards

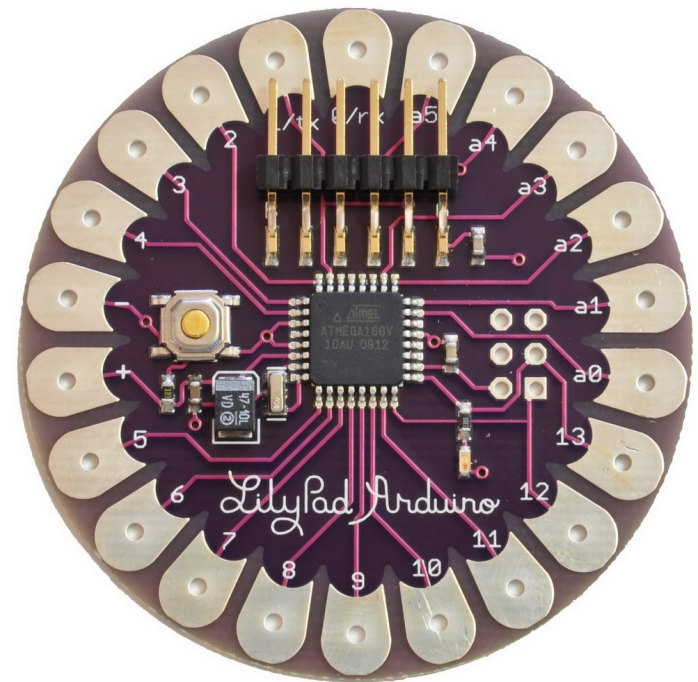
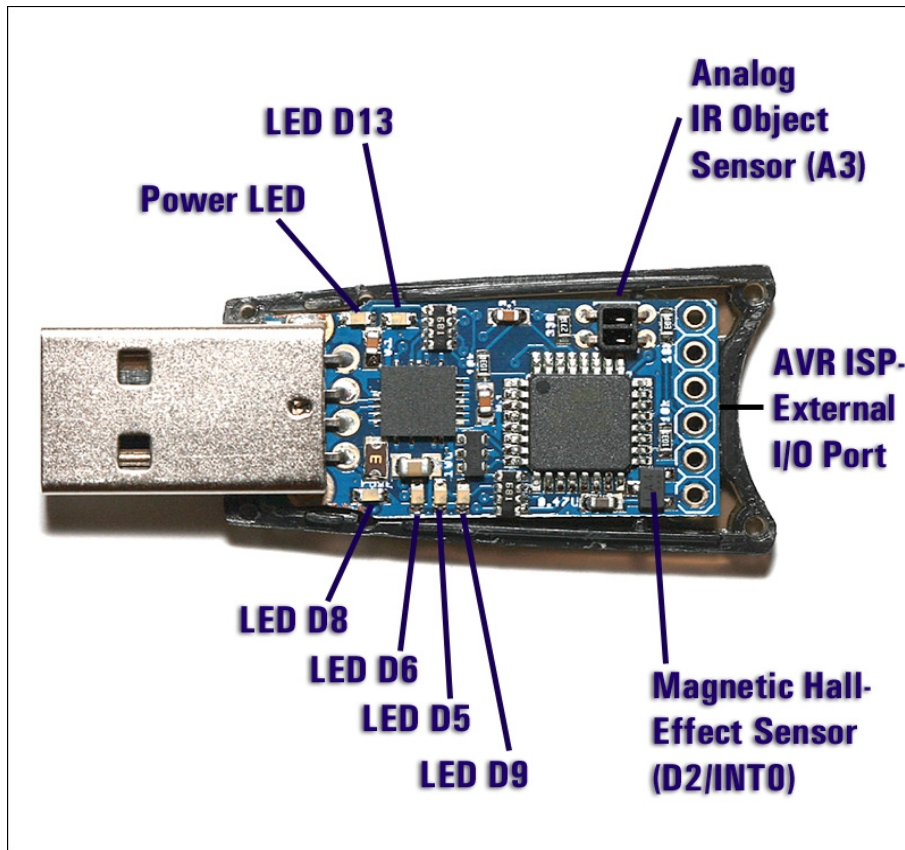


ProMicro (2012)

AtMega32U4

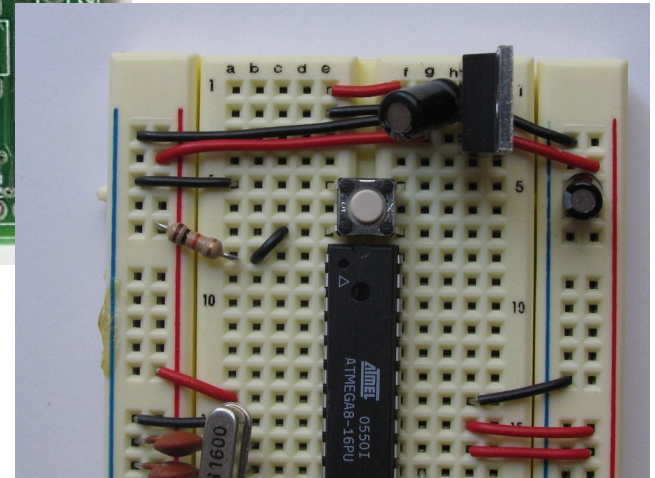
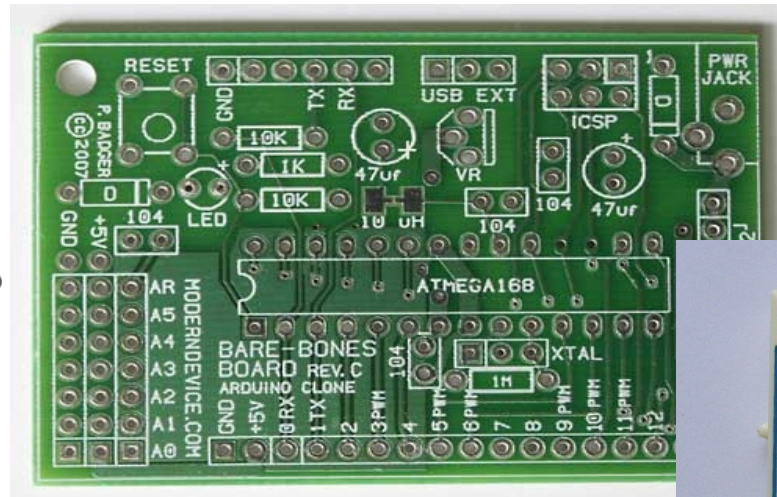
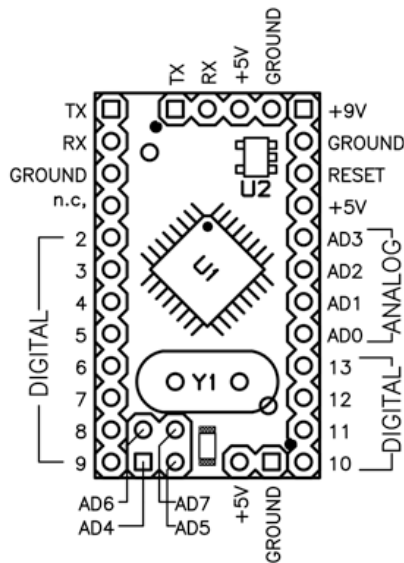
Primitive (hand made)

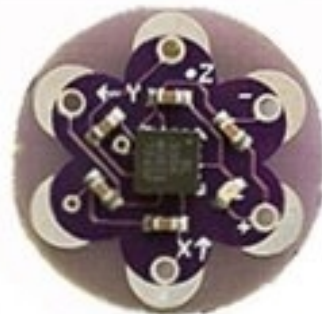
Arduino



What is Arduino?

The boards can be assembled by hand or purchased preassembled; the software can be downloaded for free





What is an “embedded system”?

What makes a microcontroller:

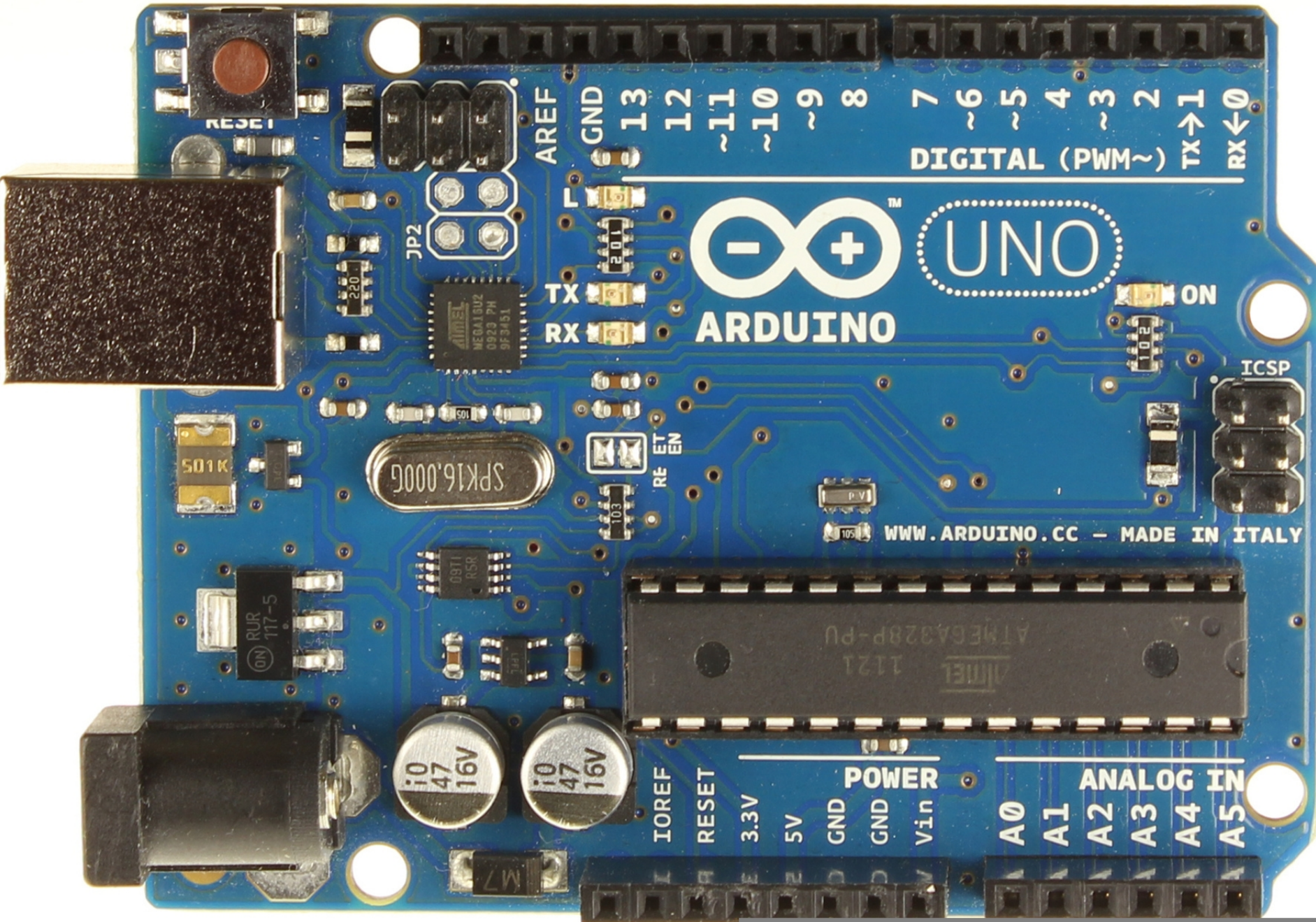
- ◆ Self Contained
 - ◆ CPU
 - ◆ Memory
 - ◆ I/O
- Application or Task Specific
 - ◆ Not a general-purpose computer
 - ◆ Appropriately scaled for the job

Designing Embedded Systems

- Microcontrollers
 - Don't have keyboard and monitor jacks
 - Must use ports to perform I/O
- Inputs - to sense things
- Outputs - to control things
- Related Component Topics
 - Cool Parts
 - Common Interfaces
 - Part Packages

Arduino Terminology

- "*sketch*" - a program you write to run on an Arduino board
- "*pin*" - an input or output connected to something. e.g. output to an LED, input from a knob.
- "*digital*" - value is either HIGH or LOW. (aka on/off, one/zero) e.g. switch state
- "*analog out*" (PWM) - value ranges, usually from 0-255. e.g. LED brightness, motor speed, etc.
- "*analog in*" - value ranges, usually from 0-1023.



ICSP
for USB interface

Reset

(I2C) SCL
(I2C) SDA

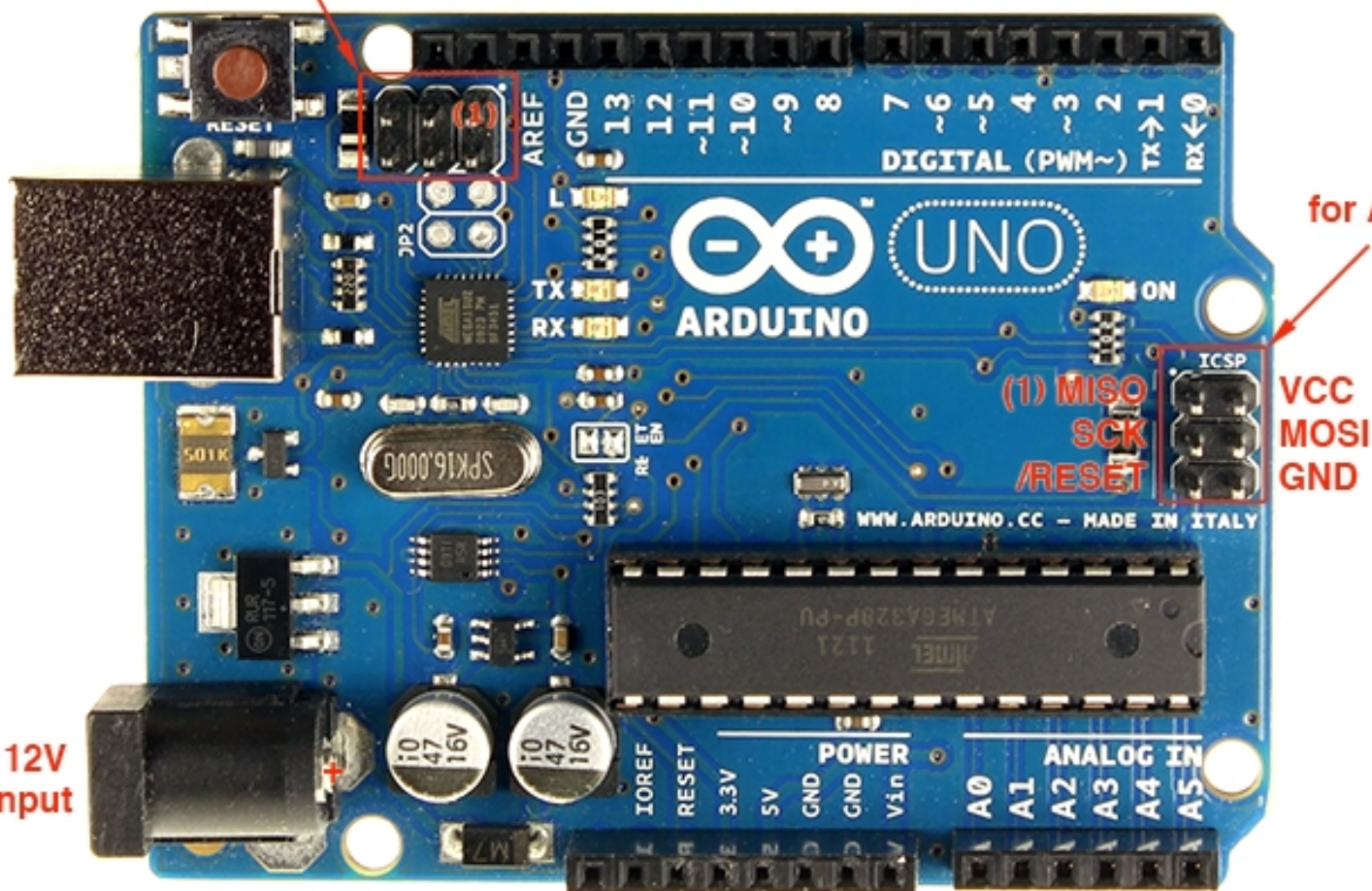
(SPI) SCK
(SPI) MISO
(SPI) MOSI
(SPI) SS

Interrupt 1
Interrupt 0

USB
to computer

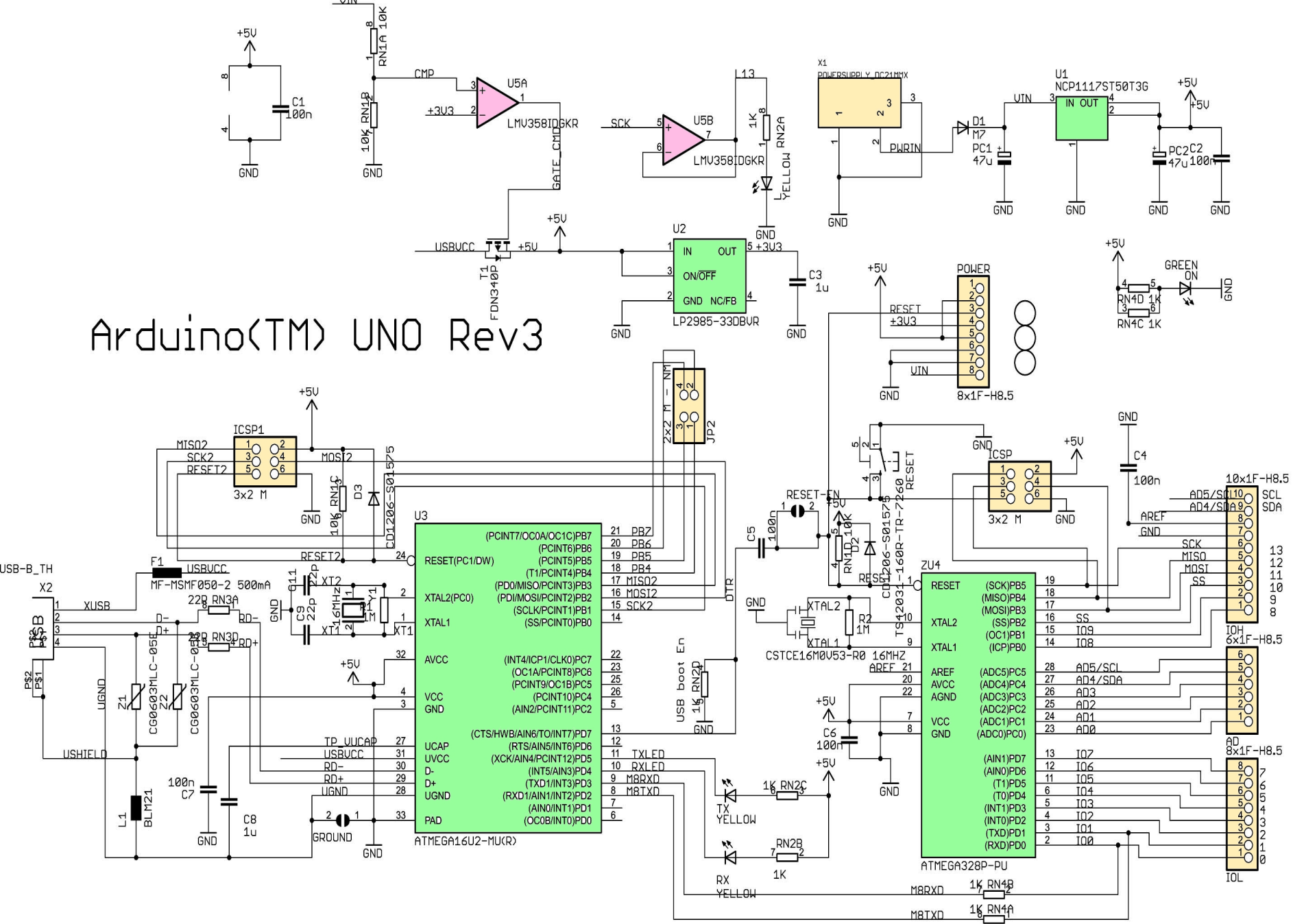
ICSP
for Atmega328

7 to 12V
DC input



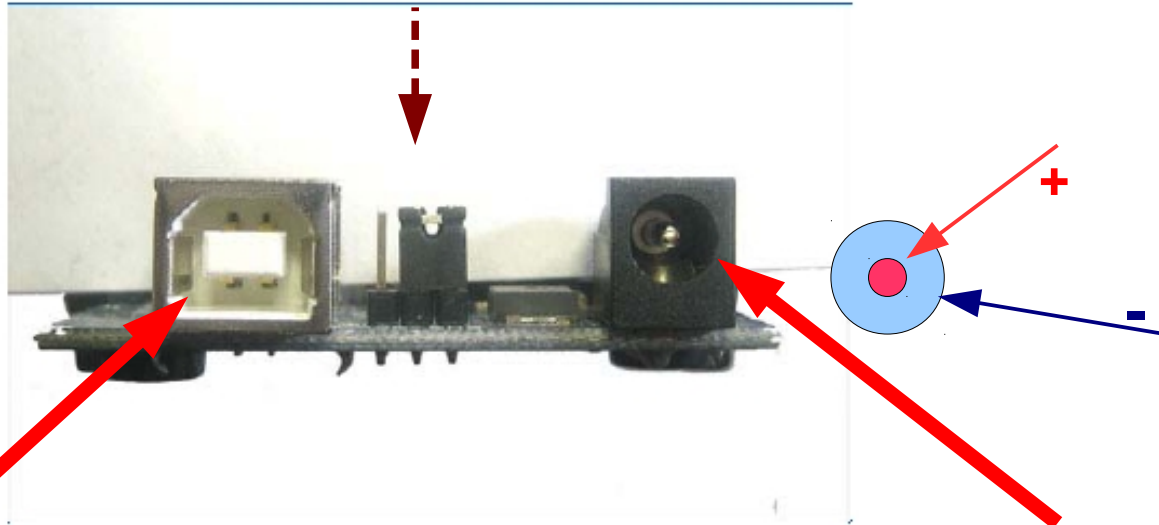
(I2C) SDA
(I2C) SCL

Arduino(TM) UNO Rev3



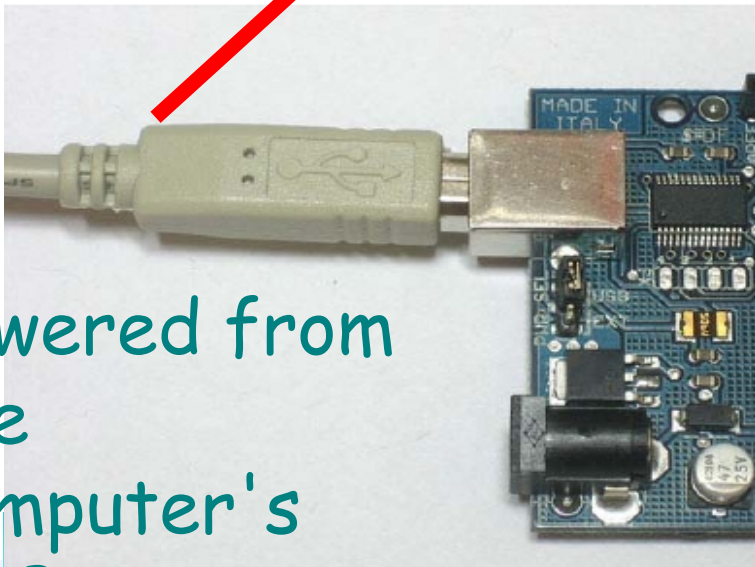
Power Supply

Jumper only on Arduino Diecimila or older



USB 5 Volt

external 7-12 Volt



powered from
the
computer's
USB port



The core: ATmega 168 (328)

- 8Bit RISC (reduced instruction set computing) Microcontroller
- AVR core RISC/modified Harvard
- 131 instructions, single level pipeline
- Up to 16/20 MIPS at 16/20MHz
- 1 instruction per clock cycle (pipelined)

Memory:

- 16KB (32KB) Flash memory; 10000 times rewritable
- 512B (1KB) EEPROM; 100000 times rewritable
- 1KB (2KB) internal SRAM

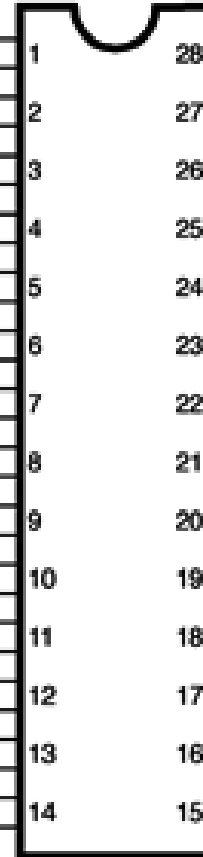


Atmega168 Pin Mapping

Arduino function

reset
digital pin 0 (RX)
digital pin 1 (TX)
digital pin 2
digital pin 3 (PWM)
digital pin 4
VCC
GND
crystal
crystal
digital pin 5 (PWM)
digital pin 6 (PWM)
digital pin 7
digital pin 8

(PCINT14/RESET) PC6
(PCINT16/RXD) PD0
(PCINT17/TXD) PD1
(PCINT18/INT0) PD2
(PCINT19/OC2B/INT1) PD3
(PCINT20/XCK/T0) PD4
VCC
GND
(PCINT6/XTAL1/TOSC1) PB6
(PCINT7/XTAL2/TOSC2) PB7
(PCINT21/OC0B/T1) PD5
(PCINT22/OC0A/AIN0) PD6
(PCINT23/AIN1) PD7
(PCINT0/CLKO/ICP1) PB0



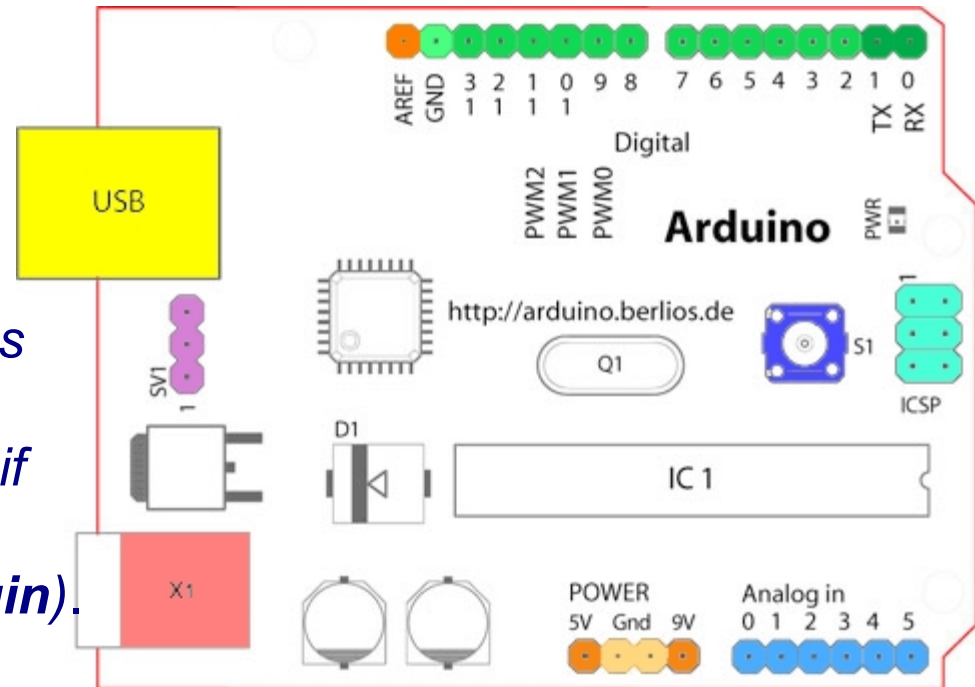
PC5 (ADC5/SCL/PCINT13)
PC4 (ADC4/SDA/PCINT12)
PC3 (ADC3/PCINT11)
PC2 (ADC2/PCINT10)
PC1 (ADC1/PCINT9)
PC0 (ADC0/PCINT8)
GND
AREF
AVCC
PB5 (SCK/PCINT5)
PB4 (MISO/PCINT4)
PB3 (MOSI/OC2A/PCINT3)
PB2 (SS/OC1B/PCINT2)
PB1 (OC1A/PCINT1)

Arduino function

analog input 5
analog input 4
analog input 3
analog input 2
analog input 1
analog input 0
GND
analog reference
VCC
digital pin 13
digital pin 12
digital pin 11 (PWM)
digital pin 10 (PWM)
digital pin 9 (PWM)

Digital Pins 11, 12 & 13 are used by the ICSP header for MISO, MOSI, SCK connections (Atmega168 pins 17, 18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

- Analog Reference pin (orange)
- Digital Ground (light green)
- Digital Pins 2-13 (green)
- Digital Pins 0-1/Serial In/Out - TX/RX (dark green) - *These pins cannot be used for digital i/o (**digitalRead** and **digitalWrite**) if you are also using serial communication (e.g. **Serial.begin**).*
- Reset Button - S1 (dark blue)
- In-circuit Serial Programmer (blue-green)
- Analog In Pins 0-5 (light blue)
- Power and Ground Pins (power: orange, grounds: light orange)
- External Power Supply In (9-12VDC) - X1 (pink)
- Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)
- USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board) (yellow)

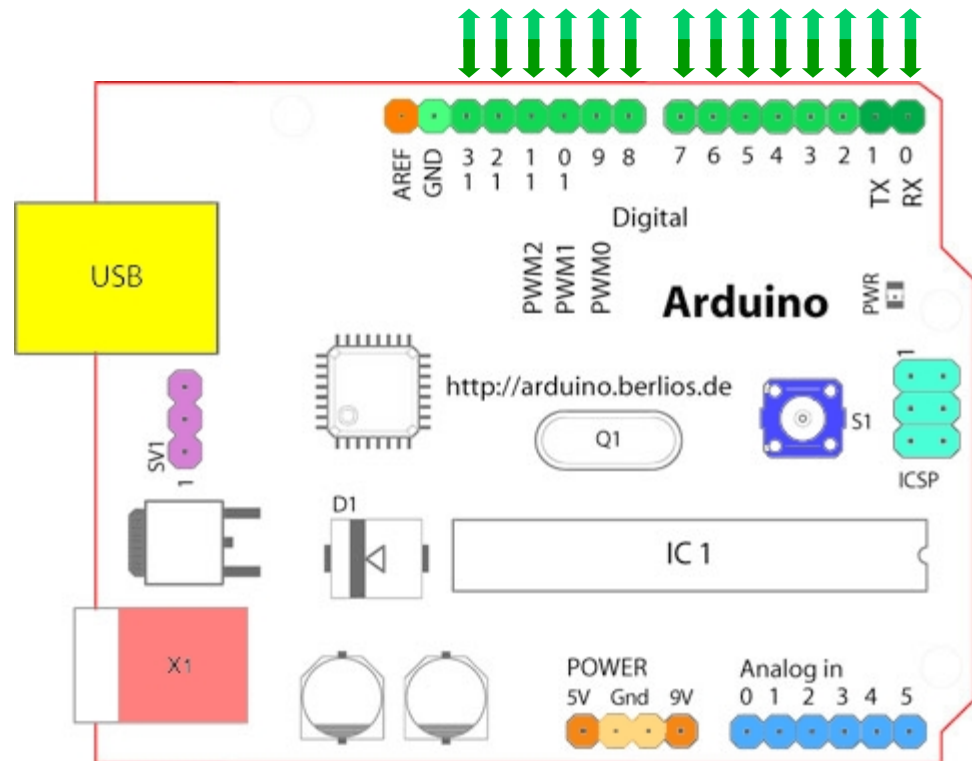


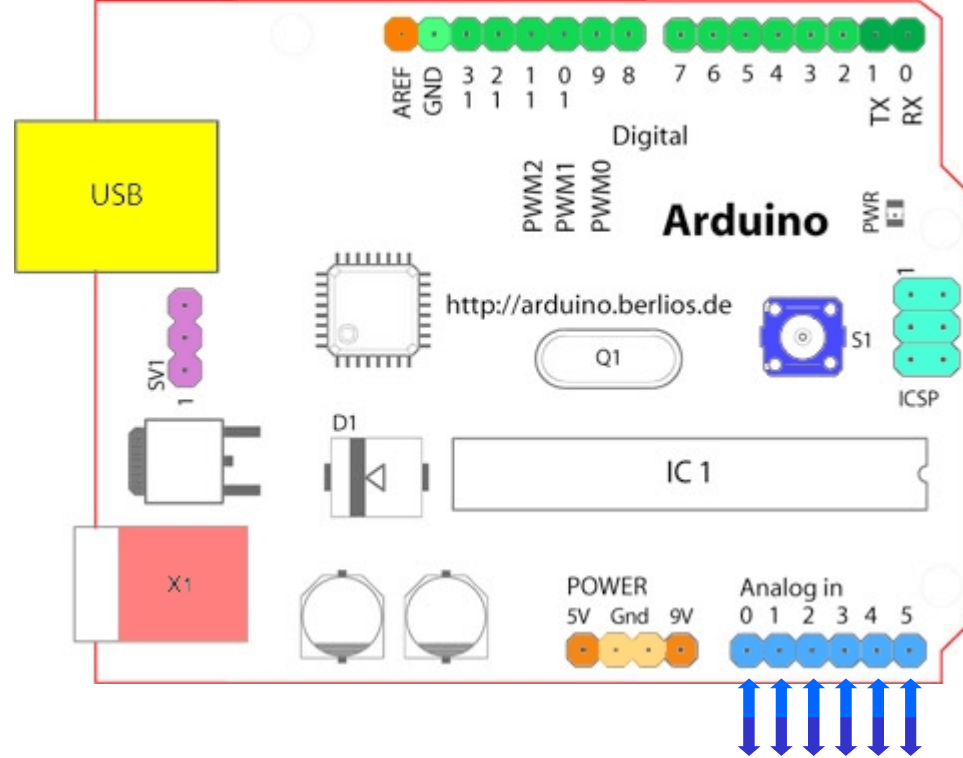
Digital I/O Pins

The Arduino I/O board has **14 Digital pins** that can be configured and used individually as Inputs or Outputs from the program. When a digital pin is configured as INPUT it can be used to read all kind of sensors that give values ON and OFF, like push buttons, touch sensors, switches etc.



When a pin is used as an OUTPUT it can be used to turn ON or OFF all sort of devices, like light bulbs, motors, home appliances etc.



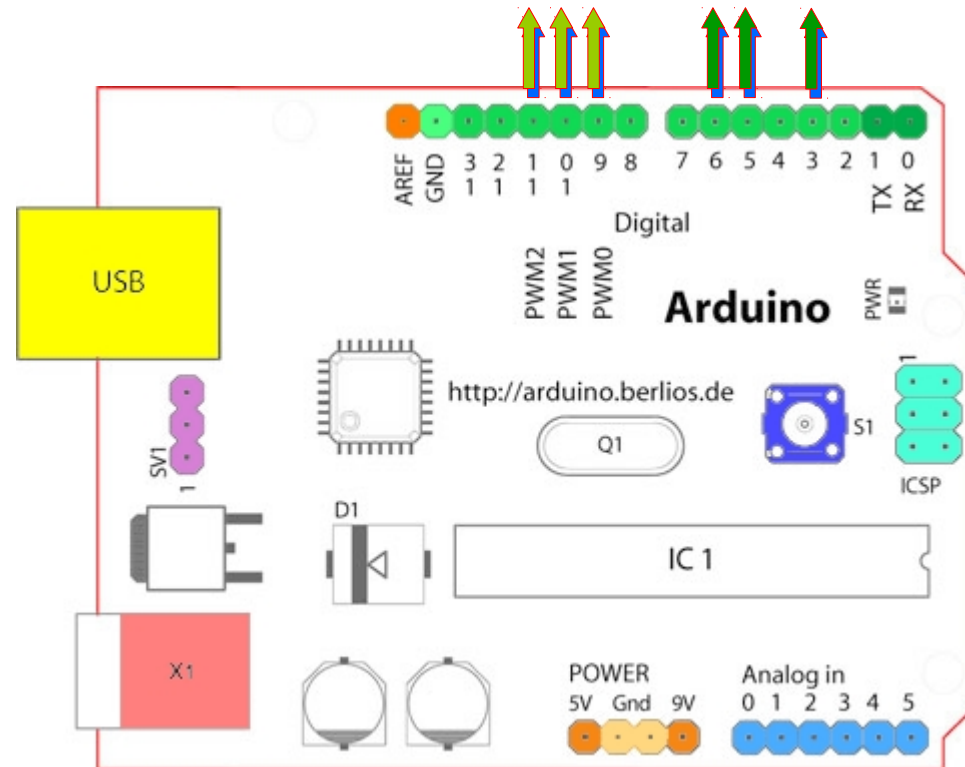


Analog Inputs

The Arduino I/O board has **6 Analog Inputs** capable of reading voltages between 0-5V. Internally the voltages are translated into number from 0 to 1023. These inputs can be used to measure continuous quantities like light intensity, temperature, proximity, position etc. depending on the type of sensor

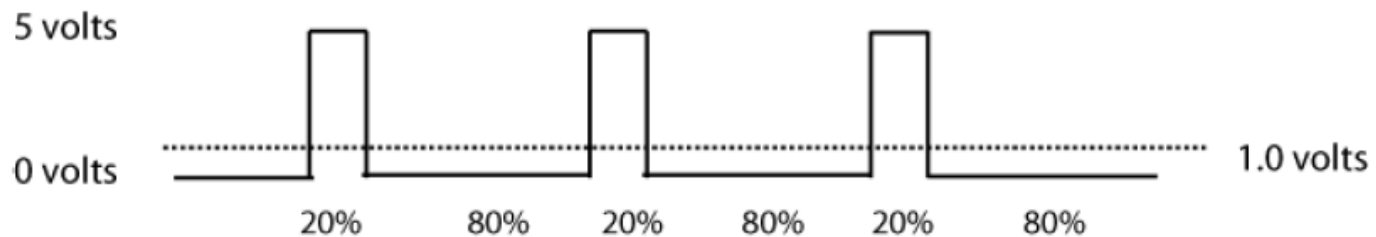
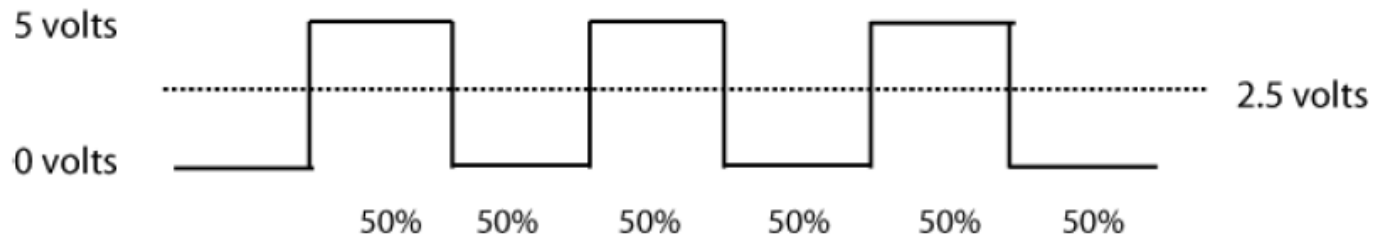
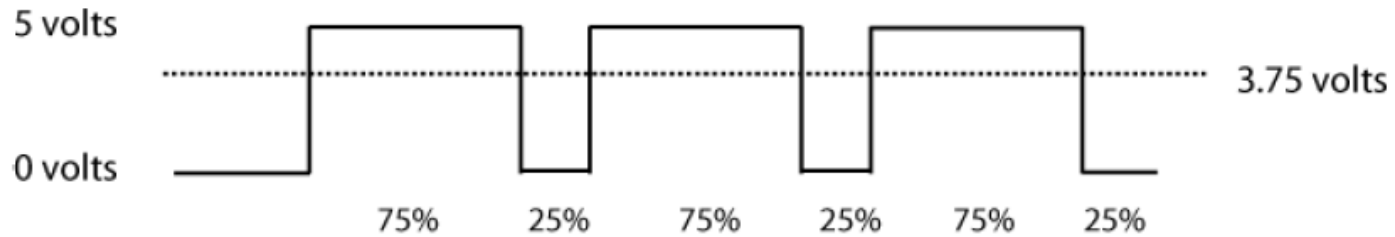
PWM (Analog) Outputs

The Arduino I/O Board has 6 PWM (Pulse Width Modulation) outputs. PWM consists of switching something ON and OFF thousands of times per second, allowing effects like dimming a light or control the speed of a motor



PWM (Analog) Outputs

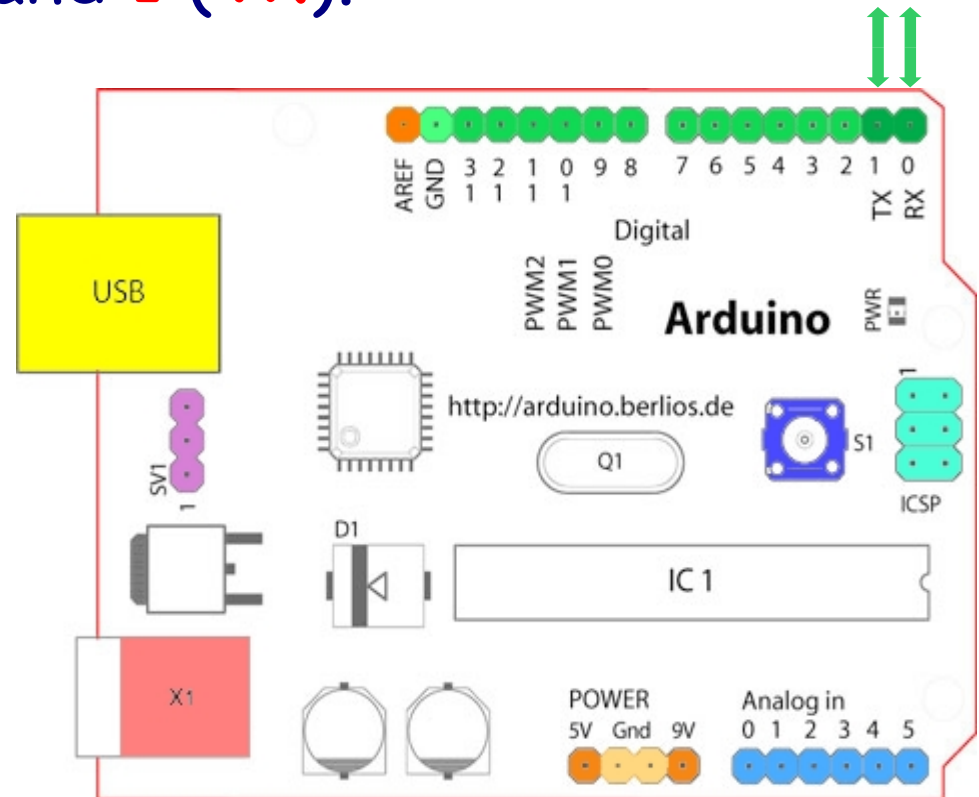
$$V_{out} = \text{MaxVoltage} * T_{on\_time} / T_{tot}$$



Serial Ports

The Arduino I/O board also has one hardware serial port. The *Serial* serial port is available on the I/O pins **0 (Rx)** and **1 (Tx)**.

It is used for both programming the Arduino I/O board by software, and to communicate with other devices

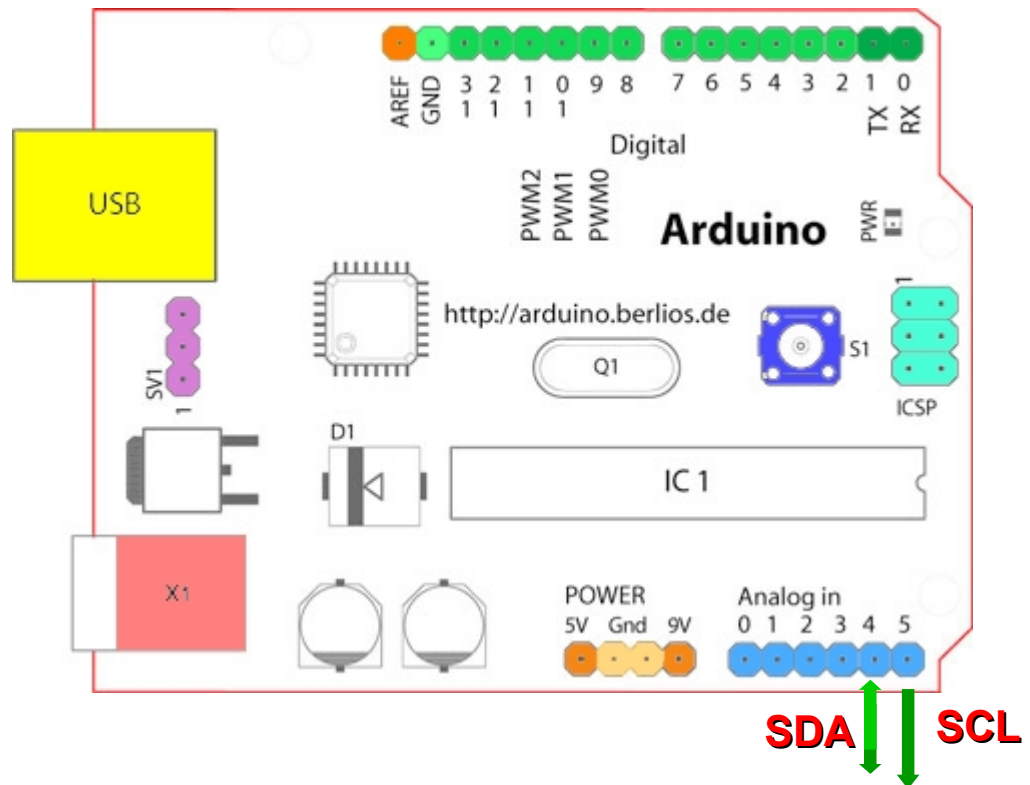


Pins With Special Functionality

The Wiring I/O board also provides **ISP** and **TWI** (Two Wire Interface) also known as **i²c interface**. The TWI allows to connect up to 128 i2c sensors/actuators in a network using only 2 pins to communicate

with all of them. It is also possible to use those pins to create a network of up to 128 Wiring I/O boards using the Wire library.

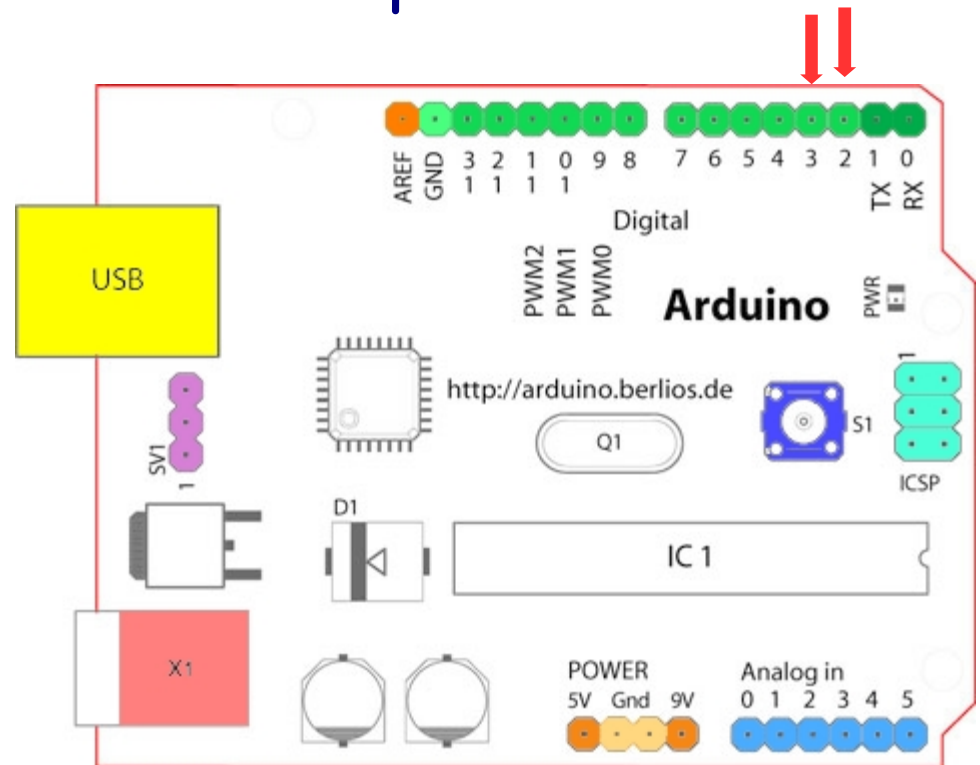
SCL = pin A5,
SDA = pin A4



External Interrupts Pins

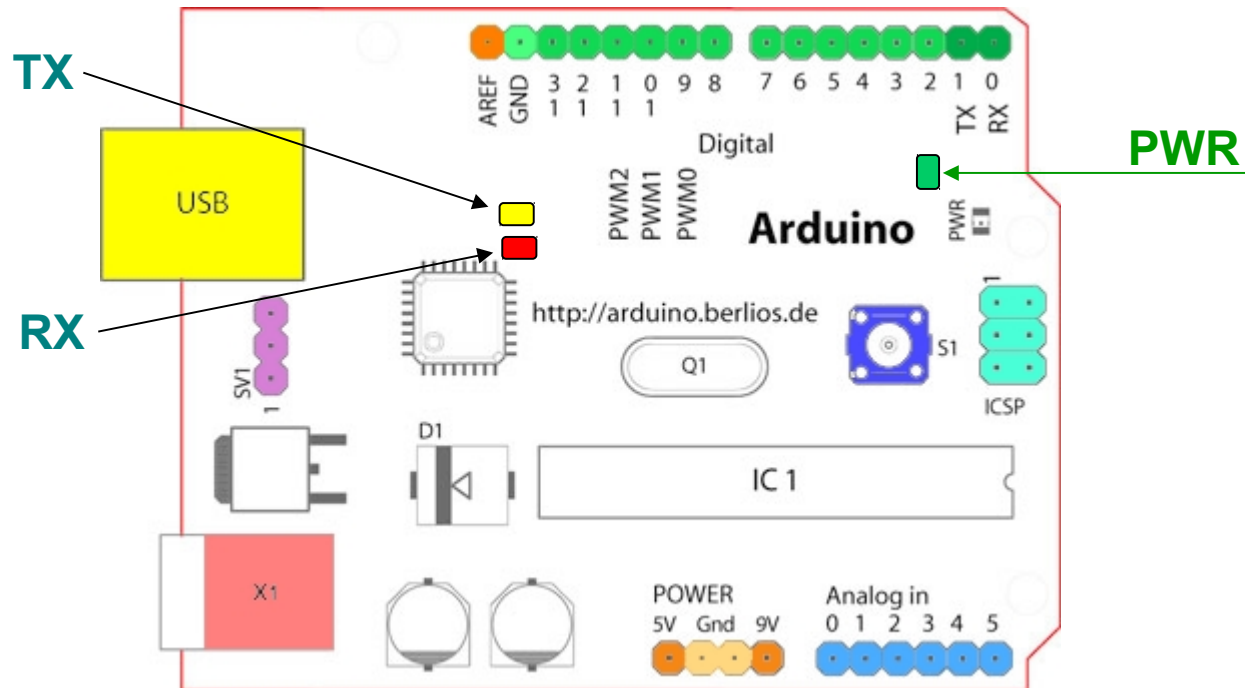
It is possible to generate and attend external interrupts on the Wiring I/O board. There are 2 external interrupts, from 0 to 1 so there are 2 pins on the Arduino I/O board capable of external interrupts, 2 and 3 respectively

In addition to being regular digital pins, note that pin 3 is also used for PWM



Indicators

The Wiring I/O board has a Power (green) LED indicator that is turned ON when the board is powered. It also includes LEDs for the Serial serial port (USB) data transmission and reception.



Arduino environment

Download the Arduino environment

To program the Arduino board you need the Arduino environment.

(<http://www.arduino.cc/files/arduino-1.xx-win.zip>)

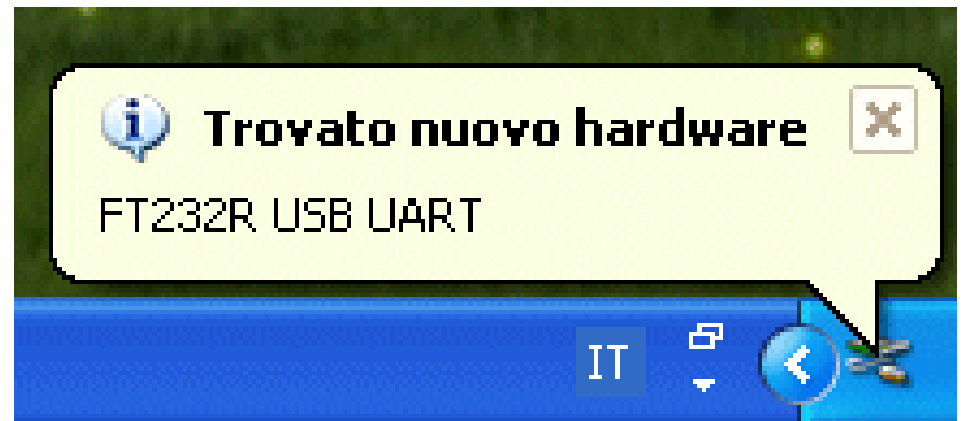
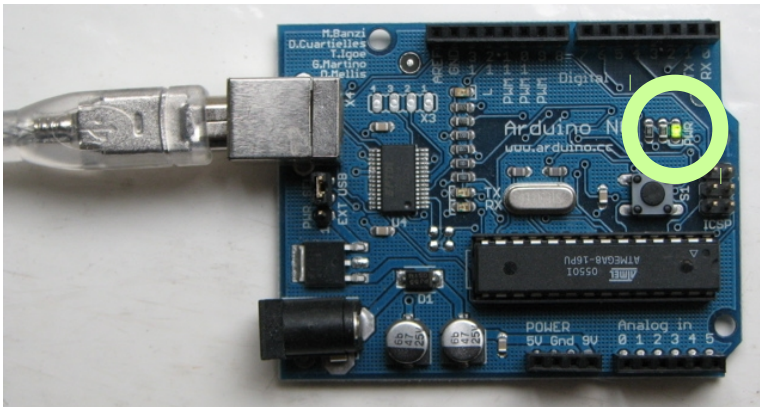
Locate the USB drivers

You will need to install the drivers (*old Arduino for the FTDI chip on the board*). These can be found in the drivers/FTDI USB Drivers directory of the Arduino distribution

Arduino environment

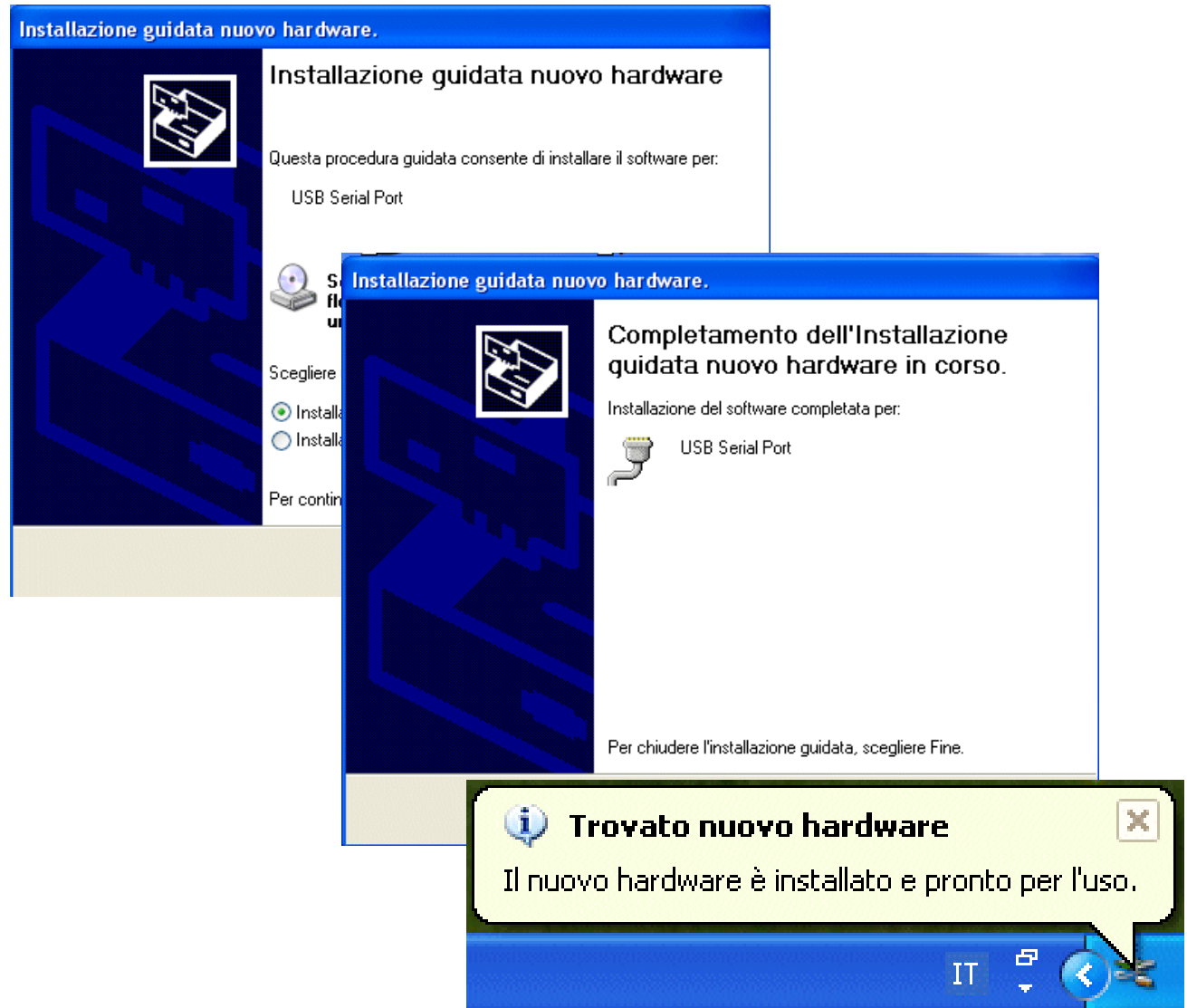
Connect the board

Connect the board to a USB port on your computer.
The power green LED [PWR] should go on .



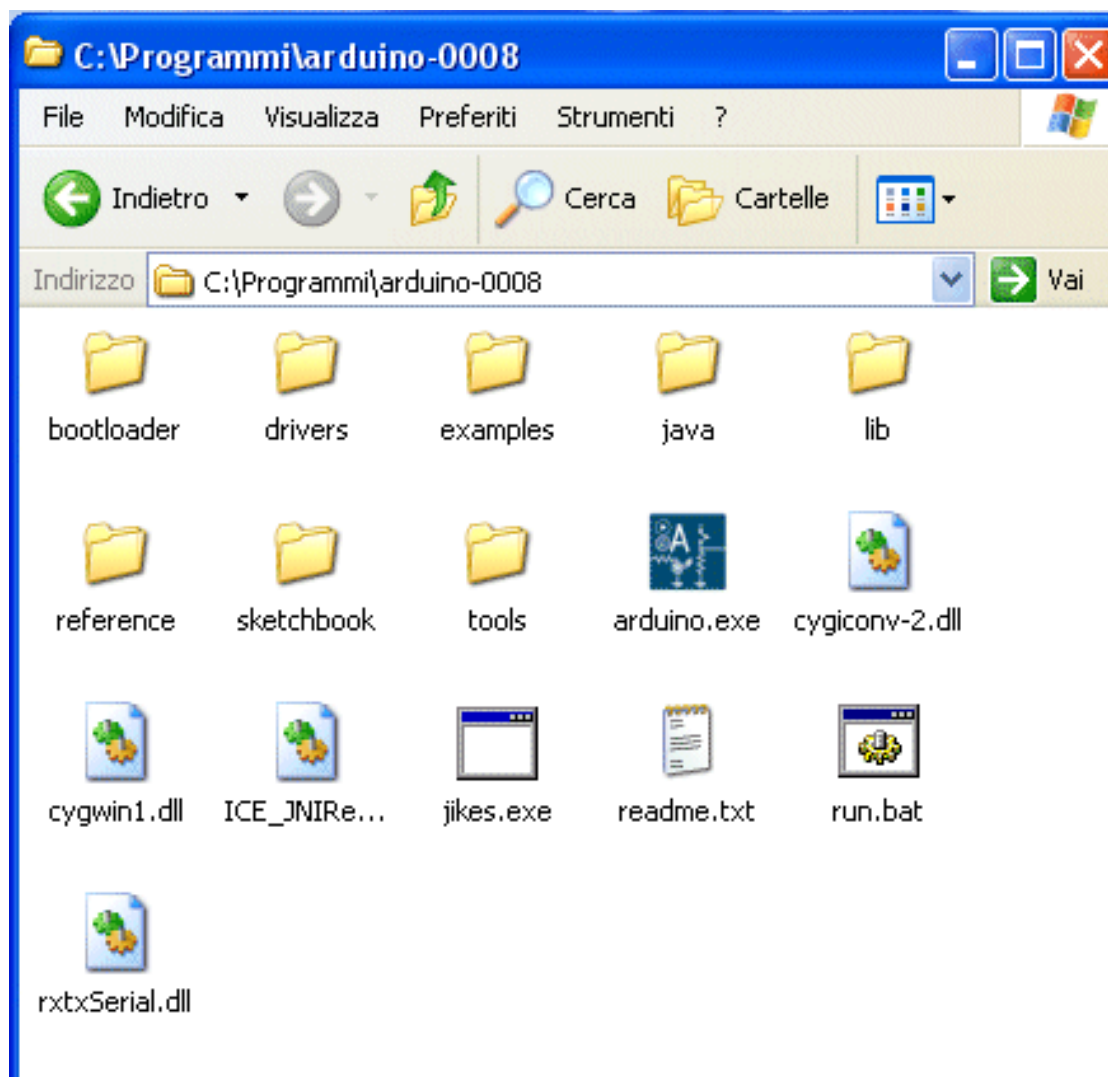
Arduino environment

The new hardware wizard will appear again. Go through the same steps. This time, a "USB Serial Port" will be found.

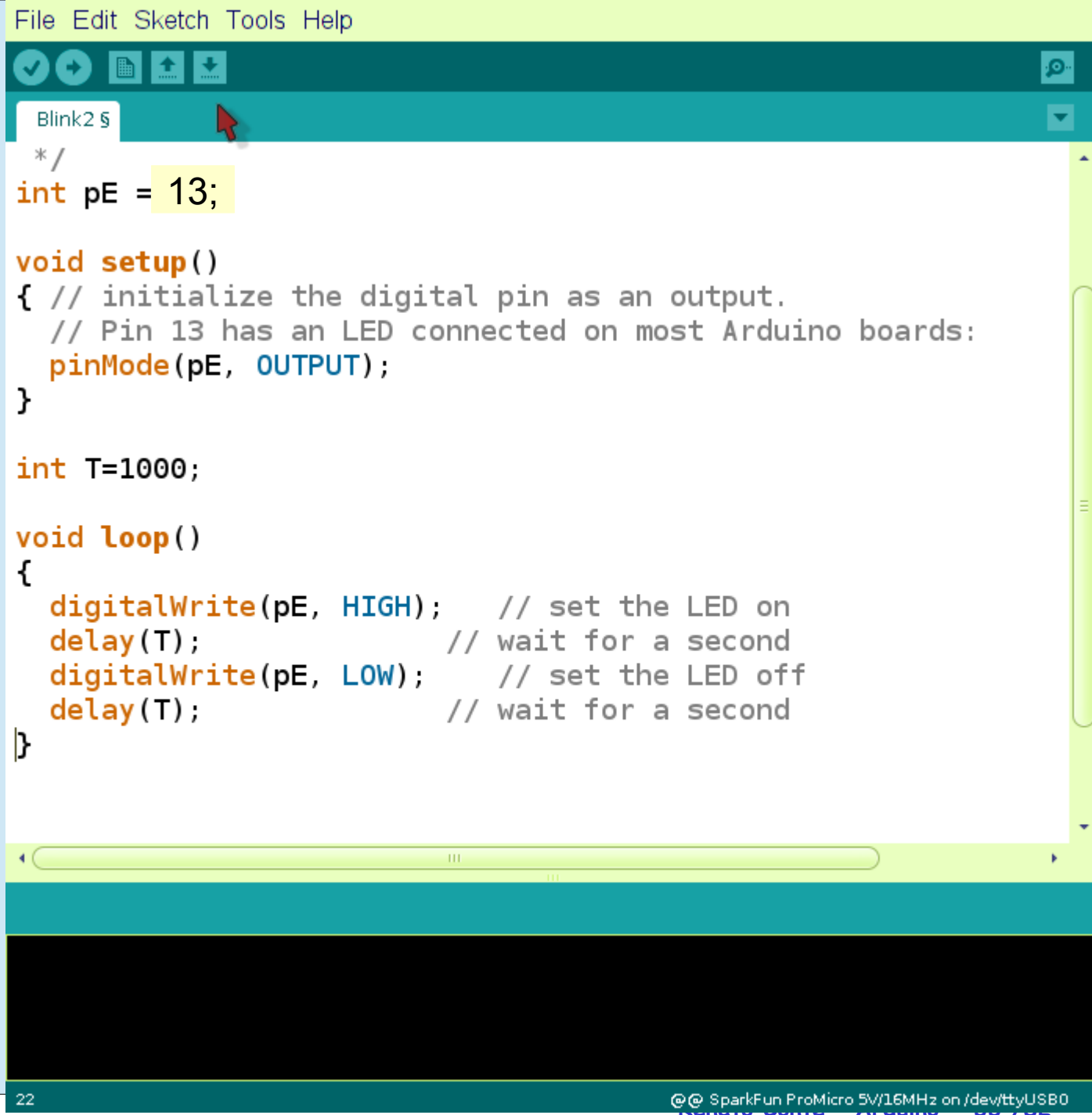


Arduino environment

Open the
Arduino folder
and double-click
the Arduino
application.



Arduino environment

A screenshot of the Arduino IDE interface. The top menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu is a toolbar with icons for checking, running, saving, and uploading. The main text area shows a sketch named 'Blink2' with the following code:

```
*/  
int pE = 13;  
  
void setup()  
{ // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino boards:  
  pinMode(pE, OUTPUT);  
}  
  
int T=1000;  
  
void loop()  
{  
  digitalWrite(pE, HIGH); // set the LED on  
  delay(T); // wait for a second  
  digitalWrite(pE, LOW); // set the LED off  
  delay(T); // wait for a second  
}
```

The IDE has a light green theme. A status bar at the bottom shows '22' on the left and 'SparkFun ProMicro 5V/16MHz on /dev/ttyUSB0' on the right.

```
/ ** *****  
@brief Gestione di un display 7 segmenti  
       come contatore dei secondi 0-9  
@author Renato Conte  
@date   10.01.2014  
#include <Arduino.h>  
#define PIN_A 1  
#define PIN_B 2  
#define PIN_C 3  
#define PIN_D 4  
#define PIN_E 5  
#define PIN_F 6  
#define PIN_G 7  
#define PIN_H 8  
#define PIN_I 9  
#define PIN_J 10  
#define PIN_K 11  
#define PIN_L 12  
#define PIN_M 13  
#define PIN_N 14  
#define PIN_O 15  
#define PIN_P 16  
#define PIN_Q 17  
#define PIN_R 18  
#define PIN_S 19  
#define PIN_T 20  
#define PIN_U 21  
#define PIN_V 22  
#define PIN_W 23  
#define PIN_X 24  
#define PIN_Y 25  
#define PIN_Z 26  
#define PIN_0 27  
#define PIN_1 28  
#define PIN_2 29  
#define PIN_3 30  
#define PIN_4 31  
#define PIN_5 32  
#define PIN_6 33  
#define PIN_7 34  
#define PIN_8 35  
#define PIN_9 36  
#define PIN_10 37  
#define PIN_11 38  
#define PIN_12 39  
#define PIN_13 40  
#define PIN_14 41  
#define PIN_15 42  
#define PIN_16 43  
#define PIN_17 44  
#define PIN_18 45  
#define PIN_19 46  
#define PIN_20 47  
#define PIN_21 48  
#define PIN_22 49  
#define PIN_23 50  
#define PIN_24 51  
#define PIN_25 52  
#define PIN_26 53  
#define PIN_27 54  
#define PIN_28 55  
#define PIN_29 56  
#define PIN_30 57  
#define PIN_31 58  
#define PIN_32 59  
#define PIN_33 60  
#define PIN_34 61  
#define PIN_35 62  
#define PIN_36 63  
#define PIN_37 64  
#define PIN_38 65  
#define PIN_39 66  
#define PIN_40 67  
#define PIN_41 68  
#define PIN_42 69  
#define PIN_43 70  
#define PIN_44 71  
#define PIN_45 72  
#define PIN_46 73  
#define PIN_47 74  
#define PIN_48 75  
#define PIN_49 76  
#define PIN_50 77  
#define PIN_51 78  
#define PIN_52 79  
#define PIN_53 80  
#define PIN_54 81  
#define PIN_55 82  
#define PIN_56 83  
#define PIN_57 84  
#define PIN_58 85  
#define PIN_59 86  
#define PIN_60 87  
#define PIN_61 88  
#define PIN_62 89  
#define PIN_63 90  
#define PIN_64 91  
#define PIN_65 92  
#define PIN_66 93  
#define PIN_67 94  
#define PIN_68 95  
#define PIN_69 96  
#define PIN_70 97  
#define PIN_71 98  
#define PIN_72 99  
#define PIN_73 100  
#define PIN_74 101  
#define PIN_75 102  
#define PIN_76 103  
#define PIN_77 104  
#define PIN_78 105  
#define PIN_79 106  
#define PIN_80 107  
#define PIN_81 108  
#define PIN_82 109  
#define PIN_83 110  
#define PIN_84 111  
#define PIN_85 112  
#define PIN_86 113  
#define PIN_87 114  
#define PIN_88 115  
#define PIN_89 116  
#define PIN_90 117  
#define PIN_91 118  
#define PIN_92 119  
#define PIN_93 120  
#define PIN_94 121  
#define PIN_95 122  
#define PIN_96 123  
#define PIN_97 124  
#define PIN_98 125  
#define PIN_99 126  
#define PIN_100 127  
#define PIN_101 128  
#define PIN_102 129  
#define PIN_103 130  
#define PIN_104 131  
#define PIN_105 132  
#define PIN_106 133  
#define PIN_107 134  
#define PIN_108 135  
#define PIN_109 136  
#define PIN_110 137  
#define PIN_111 138  
#define PIN_112 139  
#define PIN_113 140  
#define PIN_114 141  
#define PIN_115 142  
#define PIN_116 143  
#define PIN_117 144  
#define PIN_118 145  
#define PIN_119 146  
#define PIN_120 147  
#define PIN_121 148  
#define PIN_122 149  
#define PIN_123 150  
#define PIN_124 151  
#define PIN_125 152  
#define PIN_126 153  
#define PIN_127 154  
#define PIN_128 155  
#define PIN_129 156  
#define PIN_130 157  
#define PIN_131 158  
#define PIN_132 159  
#define PIN_133 160  
#define PIN_134 161  
#define PIN_135 162  
#define PIN_136 163  
#define PIN_137 164  
#define PIN_138 165  
#define PIN_139 166  
#define PIN_140 167  
#define PIN_141 168  
#define PIN_142 169  
#define PIN_143 170  
#define PIN_144 171  
#define PIN_145 172  
#define PIN_146 173  
#define PIN_147 174  
#define PIN_148 175  
#define PIN_149 176  
#define PIN_150 177  
#define PIN_151 178  
#define PIN_152 179  
#define PIN_153 180  
#define PIN_154 181  
#define PIN_155 182  
#define PIN_156 183  
#define PIN_157 184  
#define PIN_158 185  
#define PIN_159 186  
#define PIN_160 187  
#define PIN_161 188  
#define PIN_162 189  
#define PIN_163 190  
#define PIN_164 191  
#define PIN_165 192  
#define PIN_166 193  
#define PIN_167 194  
#define PIN_168 195  
#define PIN_169 196  
#define PIN_170 197  
#define PIN_171 198  
#define PIN_172 199  
#define PIN_173 200  
#define PIN_174 201  
#define PIN_175 202  
#define PIN_176 203  
#define PIN_177 204  
#define PIN_178 205  
#define PIN_179 206  
#define PIN_180 207  
#define PIN_181 208  
#define PIN_182 209  
#define PIN_183 210  
#define PIN_184 211  
#define PIN_185 212  
#define PIN_186 213  
#define PIN_187 214  
#define PIN_188 215  
#define PIN_189 216  
#define PIN_190 217  
#define PIN_191 218  
#define PIN_192 219  
#define PIN_193 220  
#define PIN_194 221  
#define PIN_195 222  
#define PIN_196 223  
#define PIN_197 224  
#define PIN_198 225  
#define PIN_199 226  
#define PIN_200 227  
#define PIN_201 228  
#define PIN_202 229  
#define PIN_203 230  
#define PIN_204 231  
#define PIN_205 232  
#define PIN_206 233  
#define PIN_207 234  
#define PIN_208 235  
#define PIN_209 236  
#define PIN_210 237  
#define PIN_211 238  
#define PIN_212 239  
#define PIN_213 240  
#define PIN_214 241  
#define PIN_215 242  
#define PIN_216 243  
#define PIN_217 244  
#define PIN_218 245  
#define PIN_219 246  
#define PIN_220 247  
#define PIN_221 248  
#define PIN_222 249  
#define PIN_223 250  
#define PIN_224 251  
#define PIN_225 252  
#define PIN_226 253  
#define PIN_227 254  
#define PIN_228 255  
#define PIN_229 256  
#define PIN_230 257  
#define PIN_231 258  
#define PIN_232 259  
#define PIN_233 260  
#define PIN_234 261  
#define PIN_235 262  
#define PIN_236 263  
#define PIN_237 264  
#define PIN_238 265  
#define PIN_239 266  
#define PIN_240 267  
#define PIN_241 268  
#define PIN_242 269  
#define PIN_243 270  
#define PIN_244 271  
#define PIN_245 272  
#define PIN_246 273  
#define PIN_247 274  
#define PIN_248 275  
#define PIN_249 276  
#define PIN_250 277  
#define PIN_251 278  
#define PIN_252 279  
#define PIN_253 280  
#define PIN_254 281  
#define PIN_255 282  
#define PIN_256 283  
#define PIN_257 284  
#define PIN_258 285  
#define PIN_259 286  
#define PIN_260 287  
#define PIN_261 288  
#define PIN_262 289  
#define PIN_263 290  
#define PIN_264 291  
#define PIN_265 292  
#define PIN_266 293  
#define PIN_267 294  
#define PIN_268 295  
#define PIN_269 296  
#define PIN_270 297  
#define PIN_271 298  
#define PIN_272 299  
#define PIN_273 300  
#define PIN_274 301  
#define PIN_275 302  
#define PIN_276 303  
#define PIN_277 304  
#define PIN_278 305  
#define PIN_279 306  
#define PIN_280 307  
#define PIN_281 308  
#define PIN_282 309  
#define PIN_283 310  
#define PIN_284 311  
#define PIN_285 312  
#define PIN_286 313  
#define PIN_287 314  
#define PIN_288 315  
#define PIN_289 316  
#define PIN_290 317  
#define PIN_291 318  
#define PIN_292 319  
#define PIN_293 320  
#define PIN_294 321  
#define PIN_295 322  
#define PIN_296 323  
#define PIN_297 324  
#define PIN_298 325  
#define PIN_299 326  
#define PIN_300 327  
#define PIN_301 328  
#define PIN_302 329  
#define PIN_303 330  
#define PIN_304 331  
#define PIN_305 3
```

**upload
to board**

Save to disk

PC console

```
void setup()
{
    for(int n=
        pinMode(n,
    }
```

```
/** @brief funzione per accendere/spegnere i 7 segmenti
    @param cifra: da rappresentare sul display
 */
```

```
void mostraCifra(int cifra) status area
{ // matrice di conversione cifra-segmenti
```

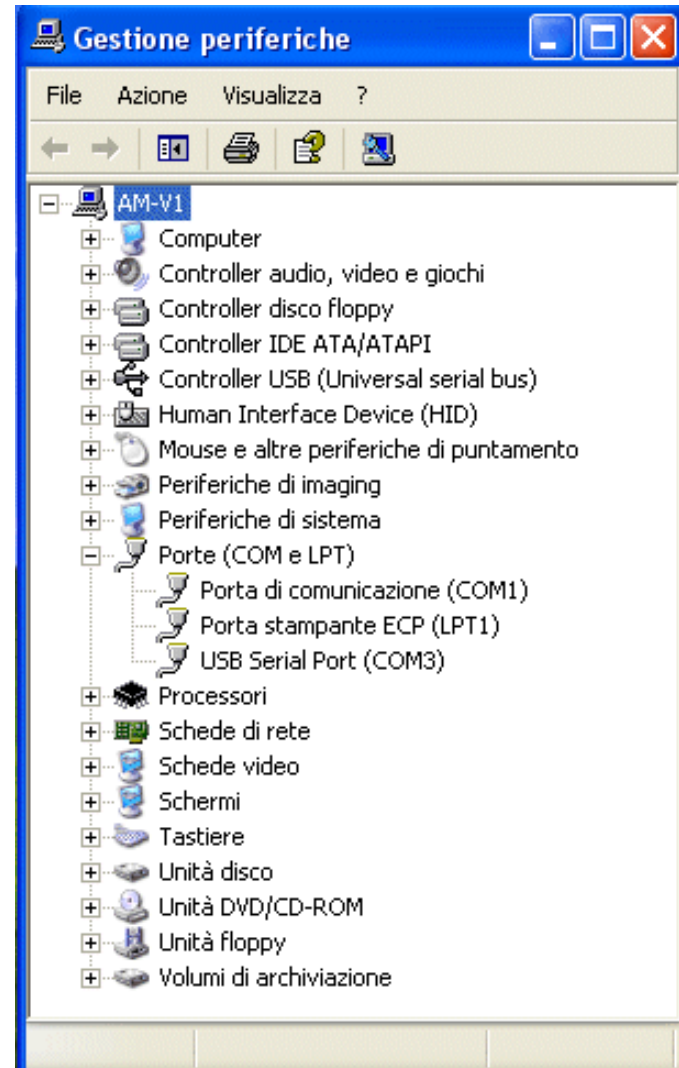
status area

Done compiling.

```
/tmp/build8234818249712731960.tmp/Display7seg.cpp.o  
/home/renato/Arduino/arduino-1.0.5/hardware/tools/avr/bin/avr-objcopy -O ihex -R .  
eeprom /tmp/build8234818249712731960.tmp/Display7seg.cpp.elf  
/tmp/build8234818249712731960.tmp/Display7seg.cpp.hex  
Binary sketch size: 1,236 bytes (of a 30,720 byte maximum)
```

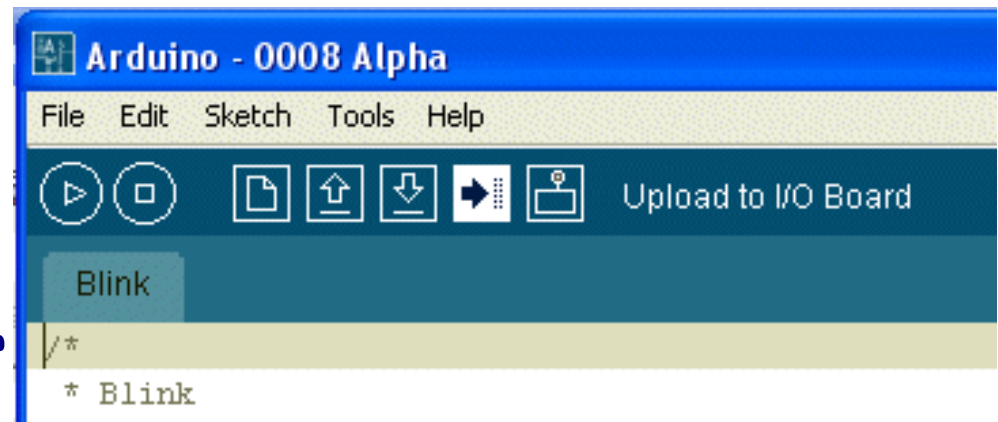
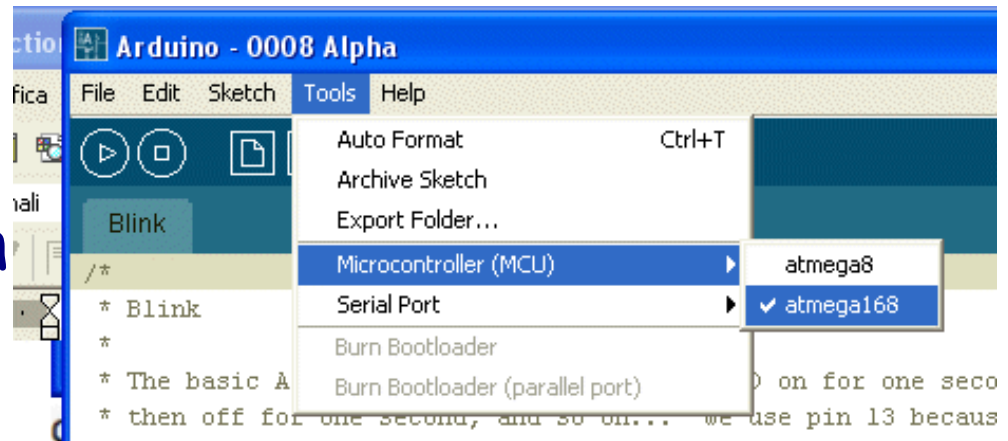
Arduino environment

Select the serial device of the Arduino board from the Tools $\Rightarrow$ Serial Port menu. On Windows, this should be COM1 or COM2 for a serial Arduino board, or COM3, COM4, or COM5 for a USB board. To find out, open the Windows Device Manager (in the Hardware tab of System control panel). Look for a "USB Serial Port" in the Ports section; that's the Arduino board.

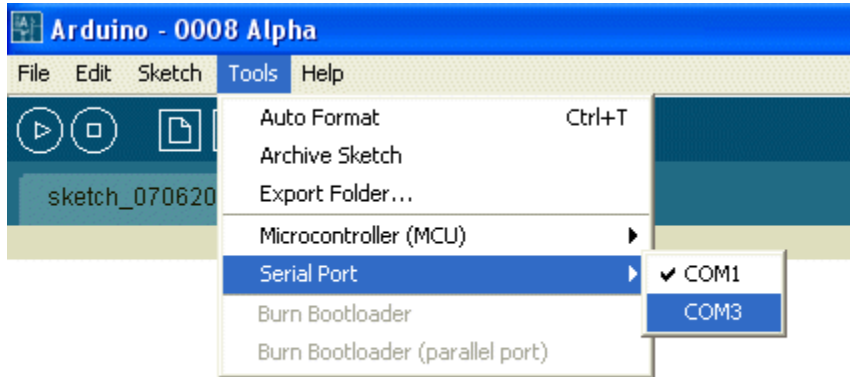


Arduino environment

You'll need to specify your microcontroller. Look at the main chip on your Arduino board. It should say either **ATmega328** or other. If the latter, you'll need to select **ATmega328** from the **Tools > Microcontroller** menu.



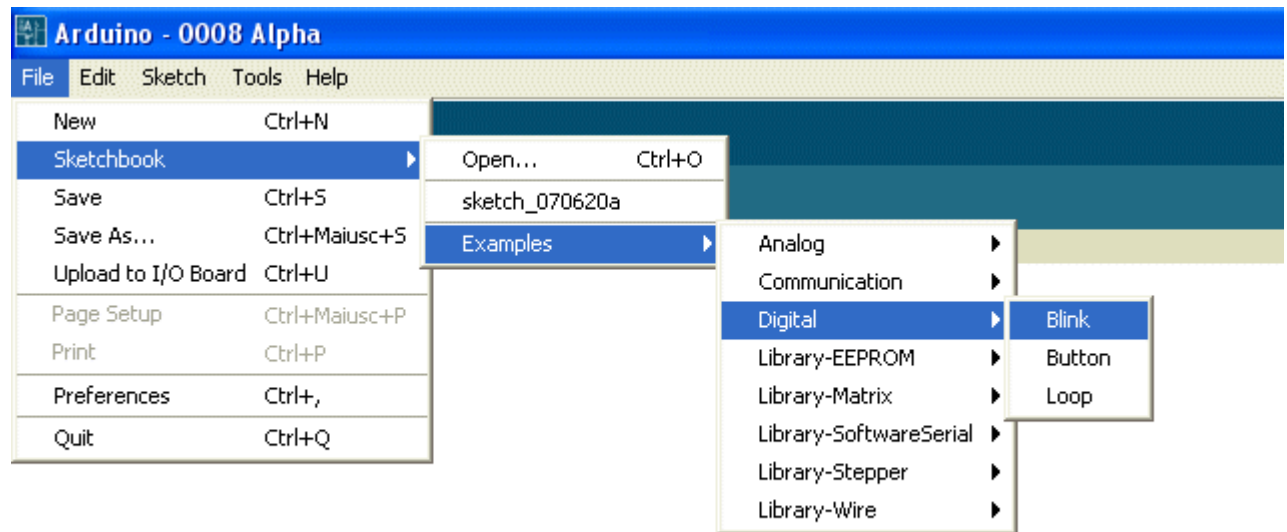
Arduino environment



Select the correct value for **SerialPort** from the **Tools** menu.

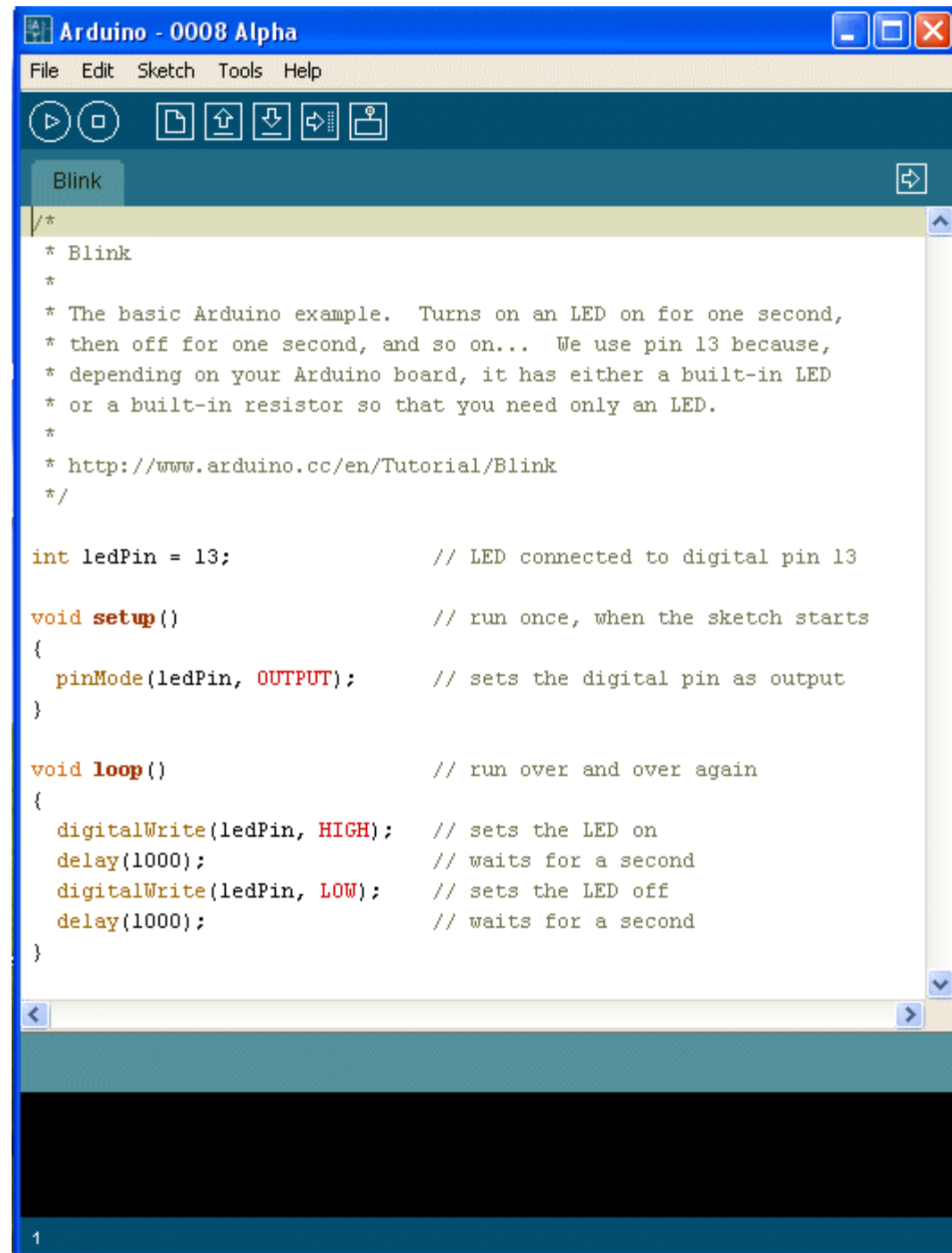
Upload a program

Open the LED blink example sketch: File > Sketchbook > Examples > Digital > Blink.



Arduino environment

The code for
the LED blink
example

A screenshot of the Arduino IDE interface. The title bar reads "Arduino - 0008 Alpha". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for running, stopping, saving, uploading, and downloading. The "Blink" tab is selected. The code editor shows the following text:

```
/*
 * Blink
 *
 * The basic Arduino example. Turns on an LED on for one second,
 * then off for one second, and so on... We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

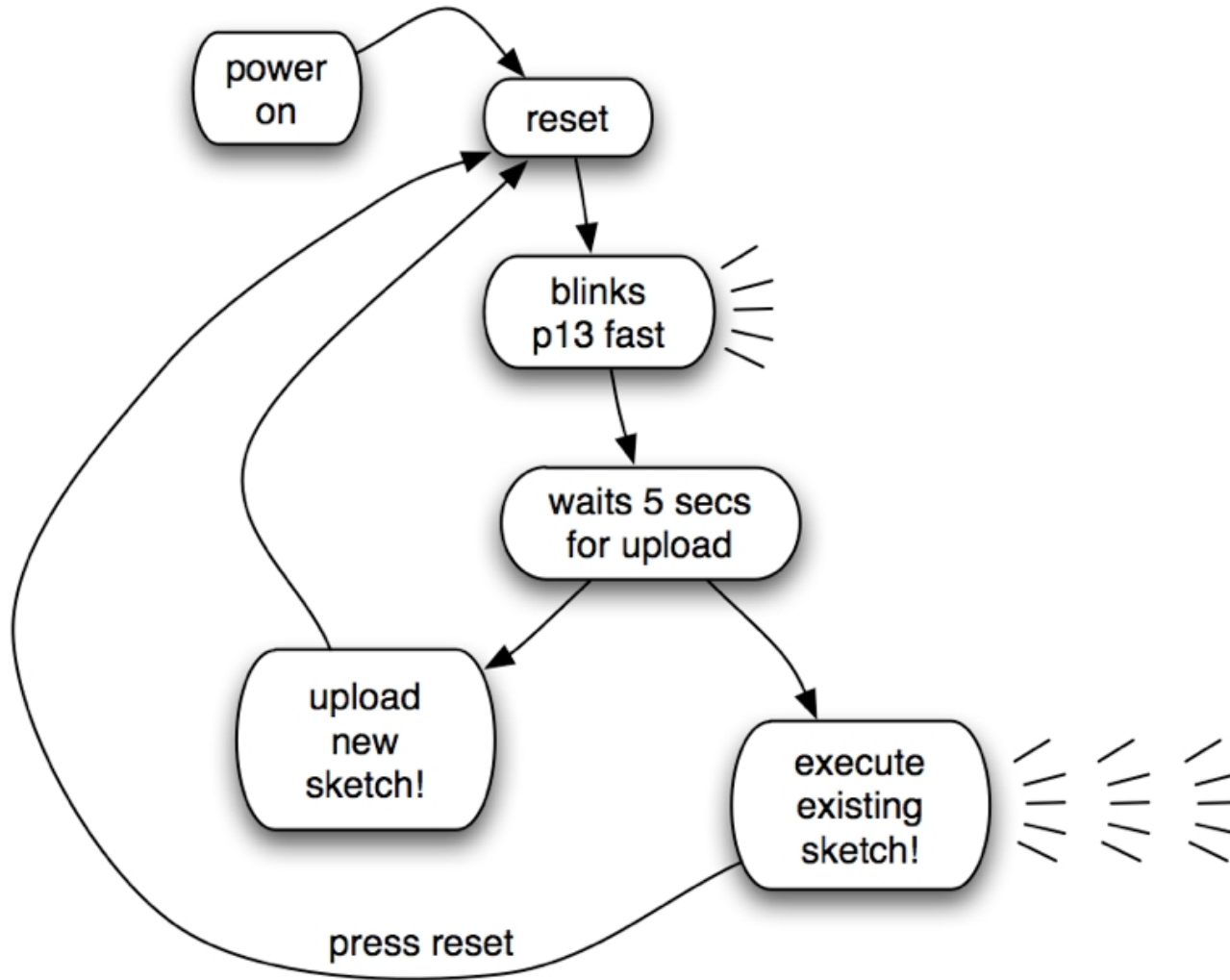
int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

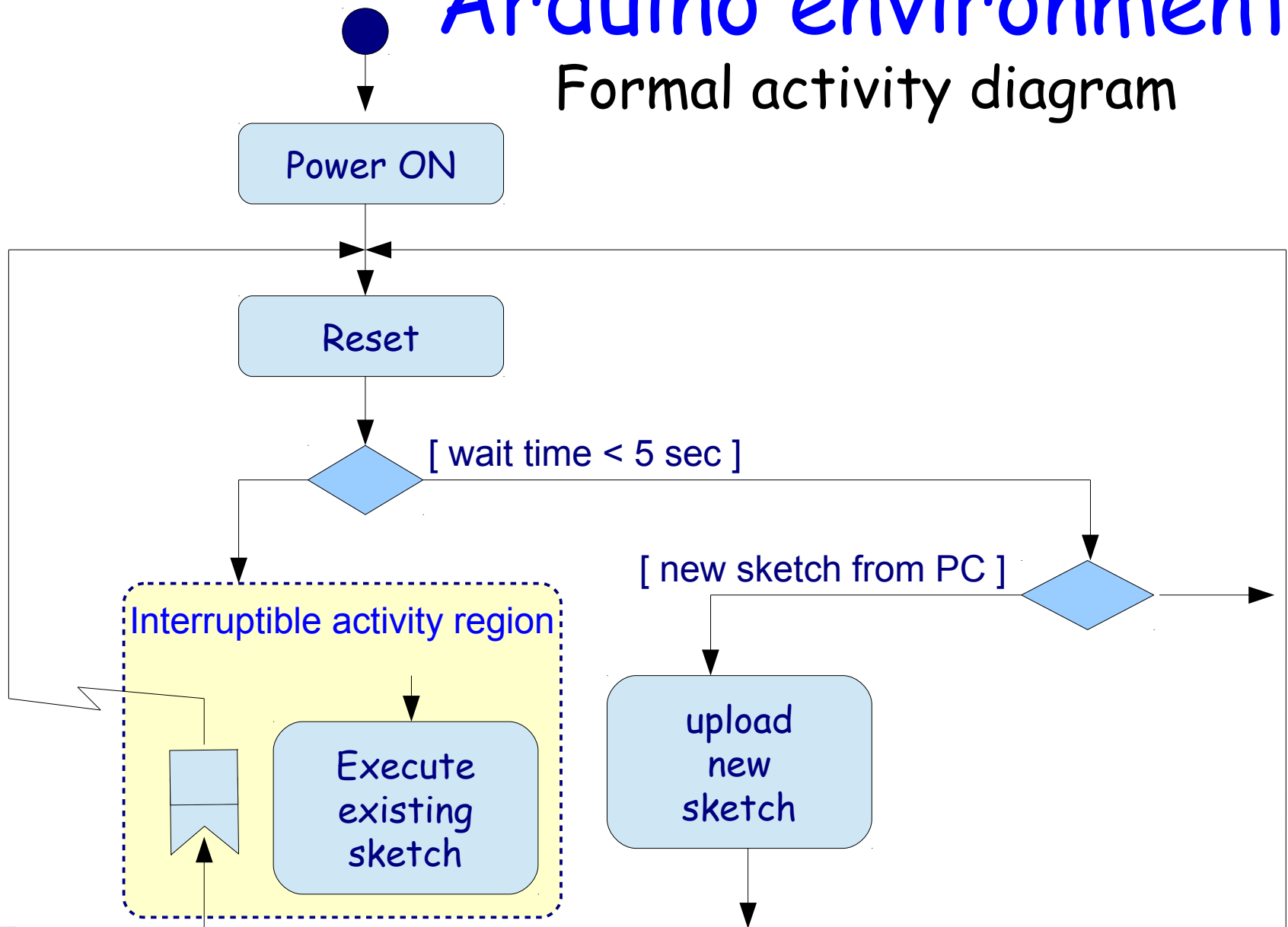
The bottom of the window shows a black area for serial output and a status bar with the number "1".

Arduino environment



Arduino environment

Formal activity diagram

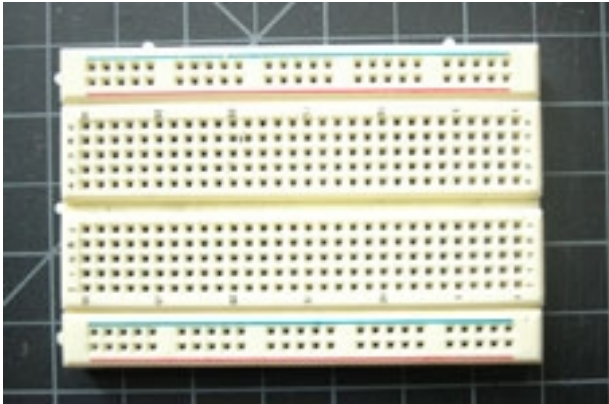


Special Interrupt
request: reset hardware

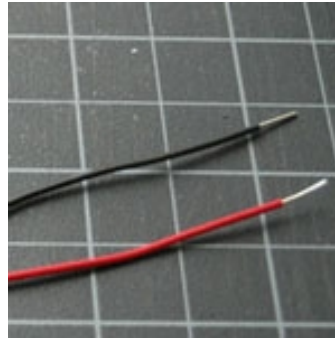
LAB: Digital In Out

In this lab, you'll connect a digital input circuit and a digital output circuit to a microcontroller. Though this is written for the Arduino microcontroller module, the principles apply to any microcontroller.

for this lab you'll need:



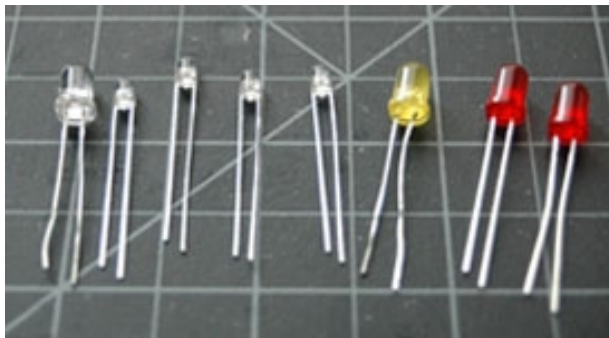
Solderless breadboard



22-AWG
hookup wire



Arduino module



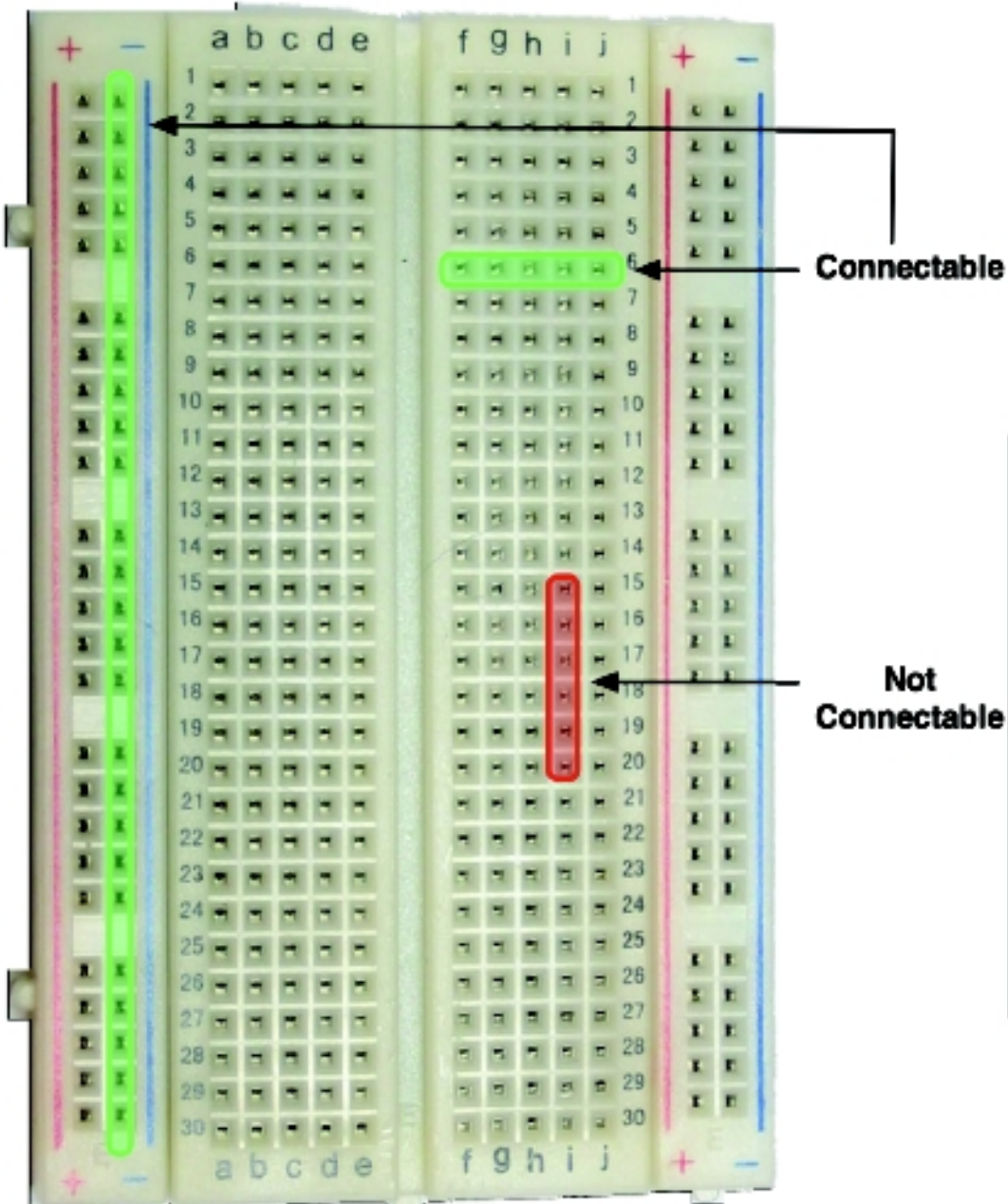
LEDs



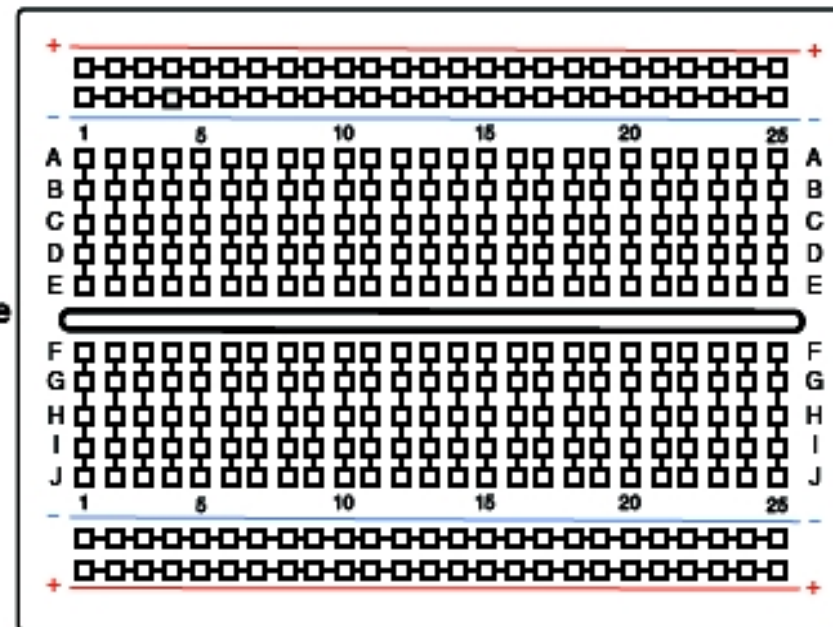
220-ohm and 10Kohm resistors



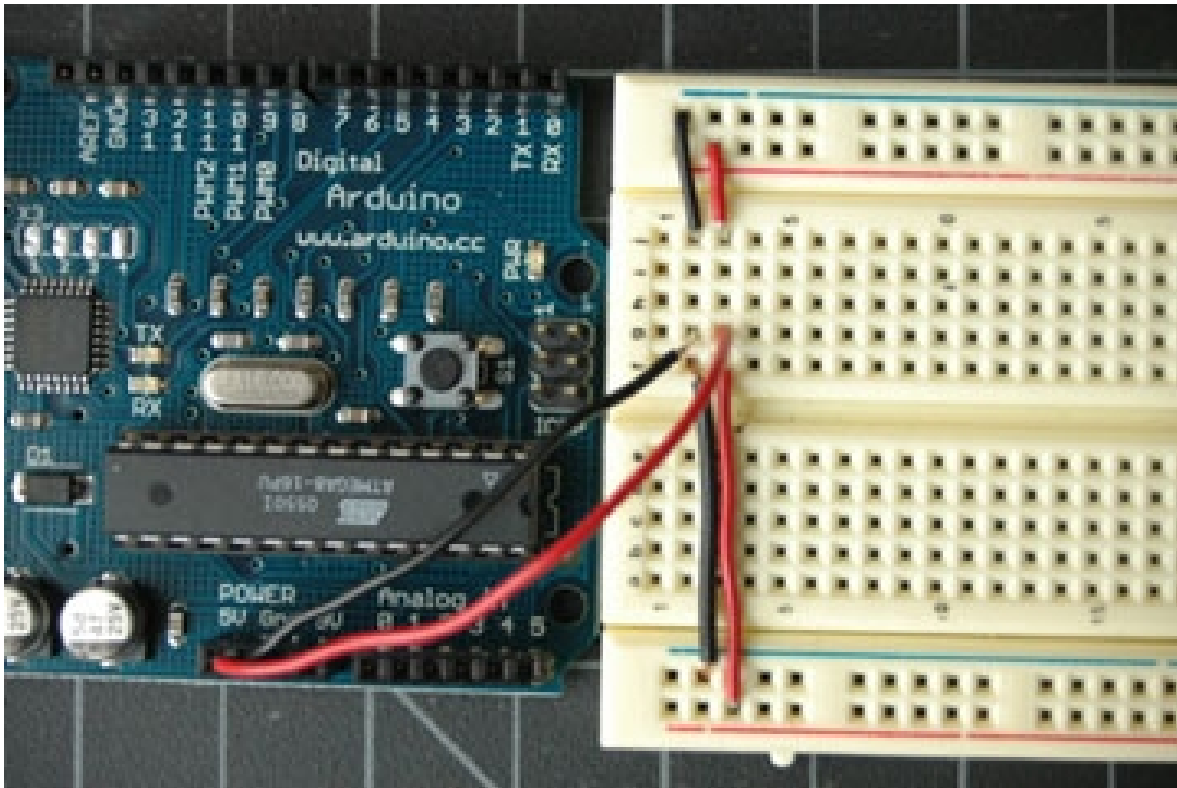
switch



Solderless Breadboard

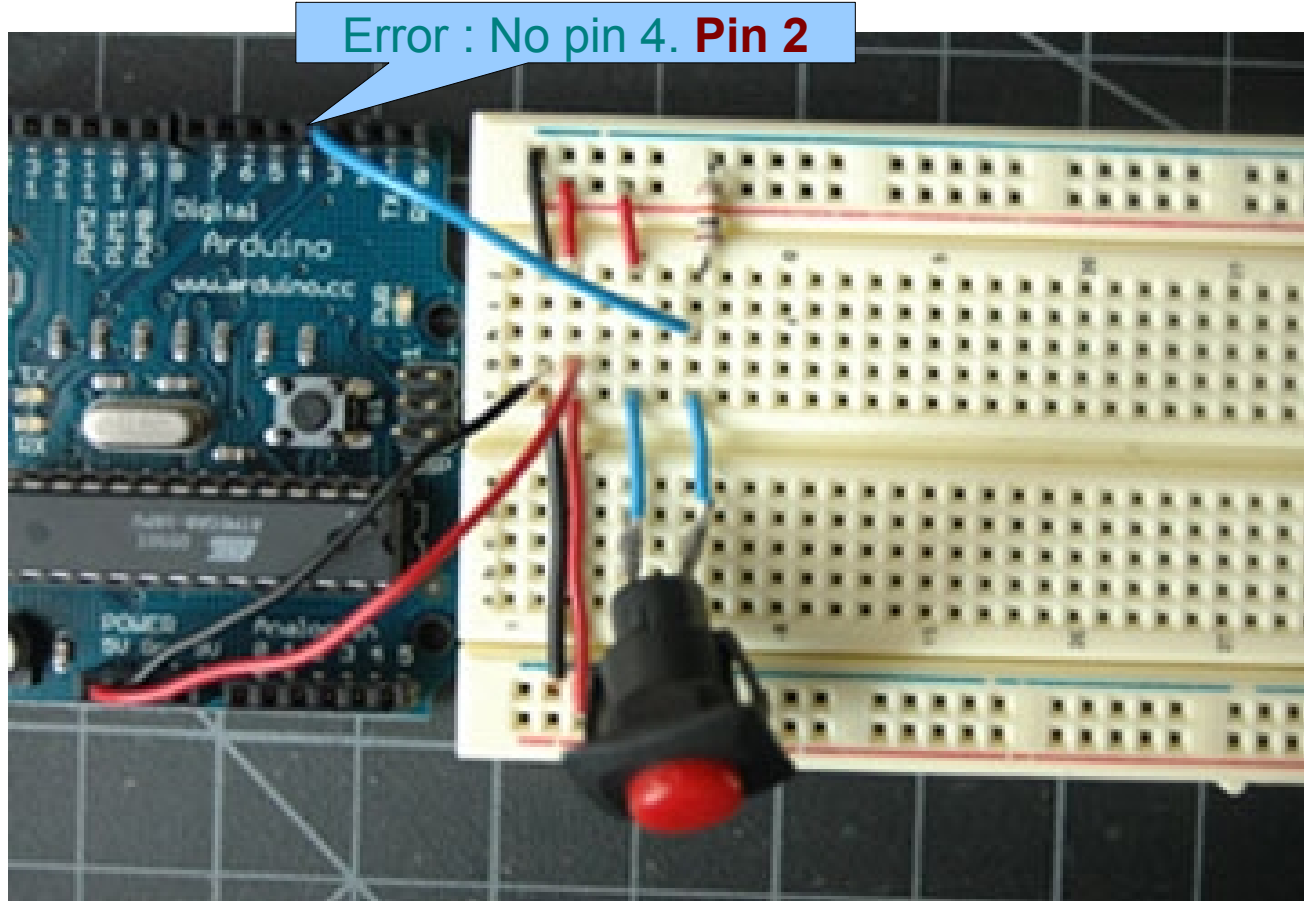


Prepare the breadboard



Connect power and ground on the breadboard to power and ground from the microcontroller. On the Arduino module, use the 5V and any of the ground connections

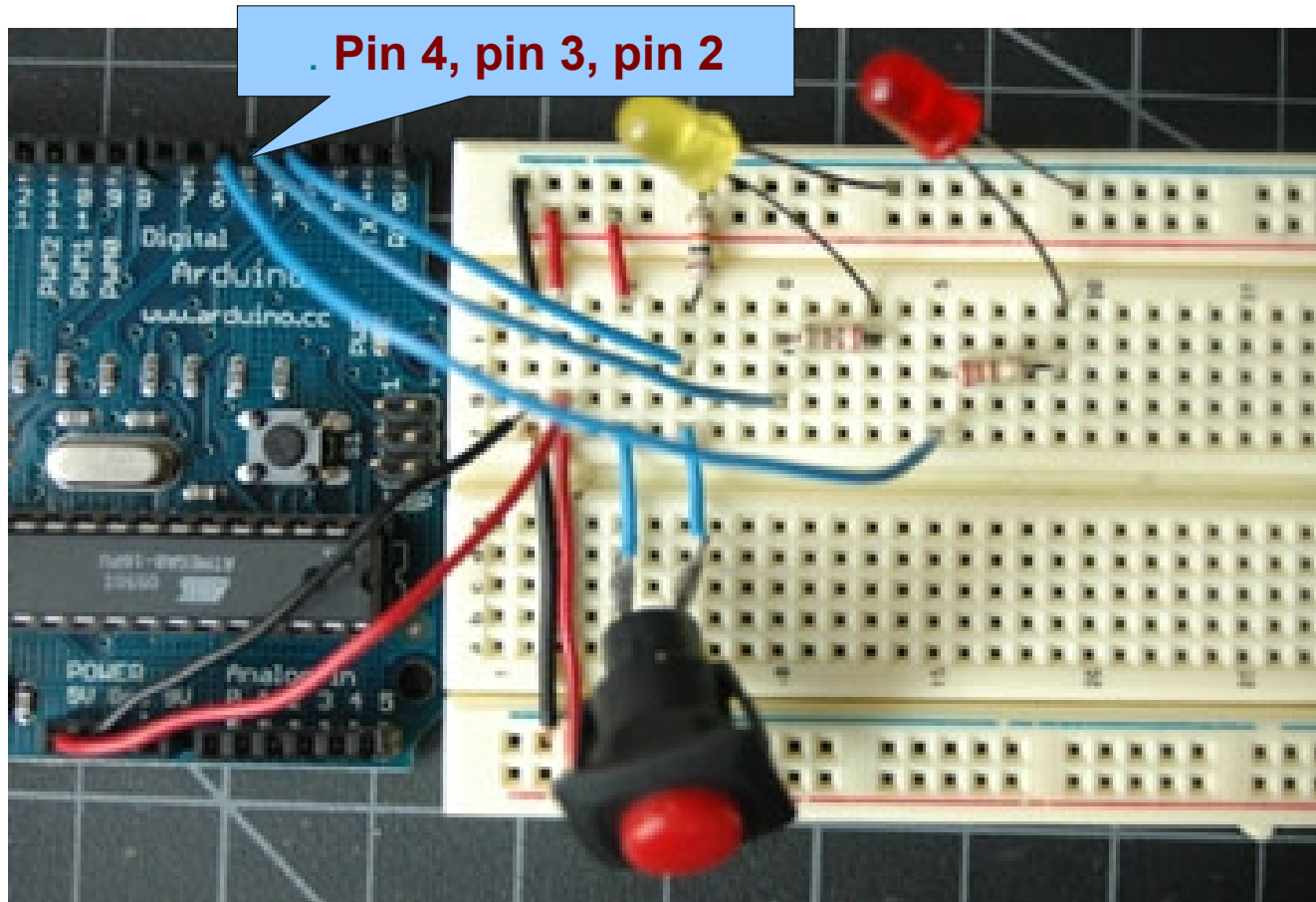
Add a Digital Input (a switch)



Connect a switch to digital input 2 on the Arduino. The switch shown below is a store-bought

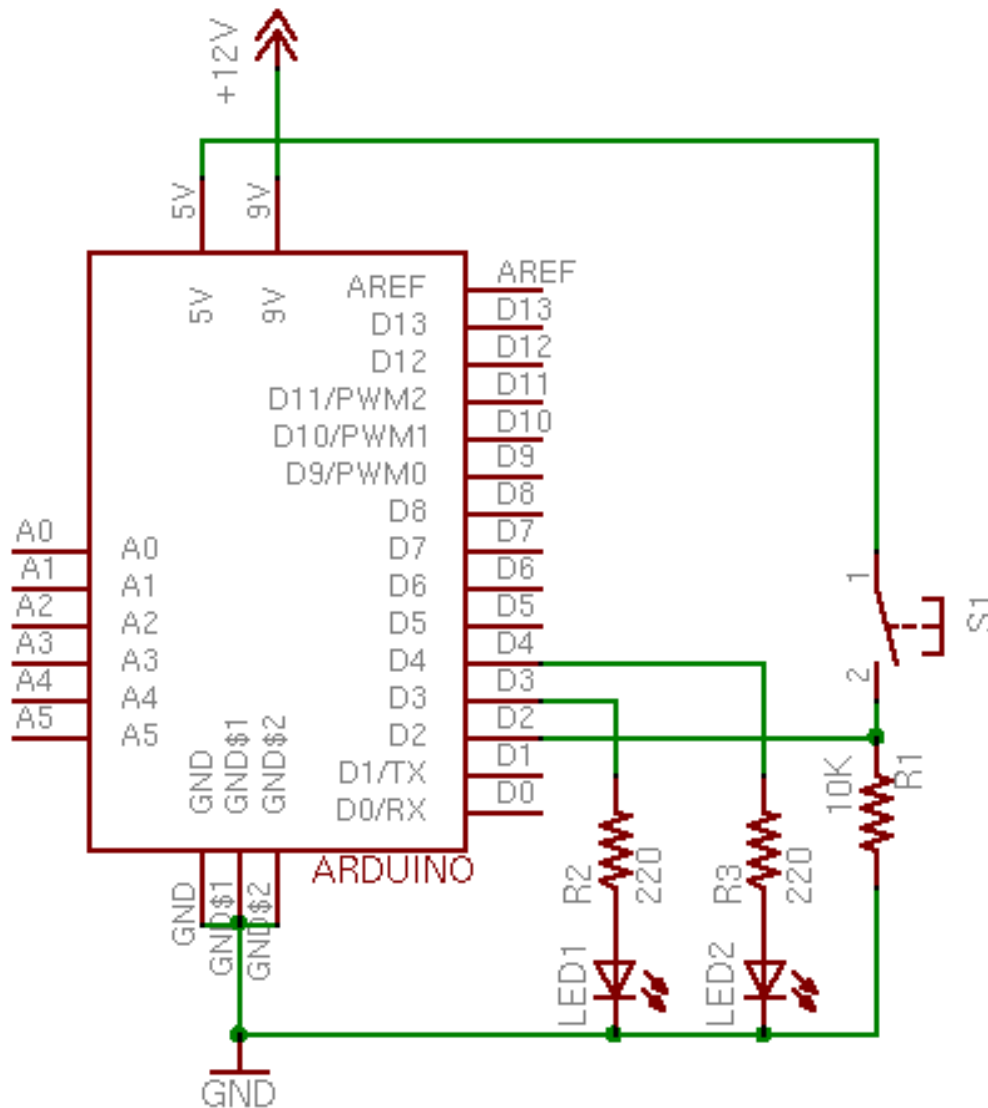
momentary pushbutton, but you can use any switch. Try making your own with a couple of pieces of metal.

Add a Digital Output (LEDs)



Connect a 220-ohm resistor and an LED in series to digital pin 3 and another to digital pin 4 of the Arduino.

Schematic



Prepare a program that reads the digital input on pin 2. Then it turns on only the LED on pin 3 if the input is high (i.e. the switch is on), or turns on only the LED on pin 4 if the input is low (the switch is off):

// declare variables:

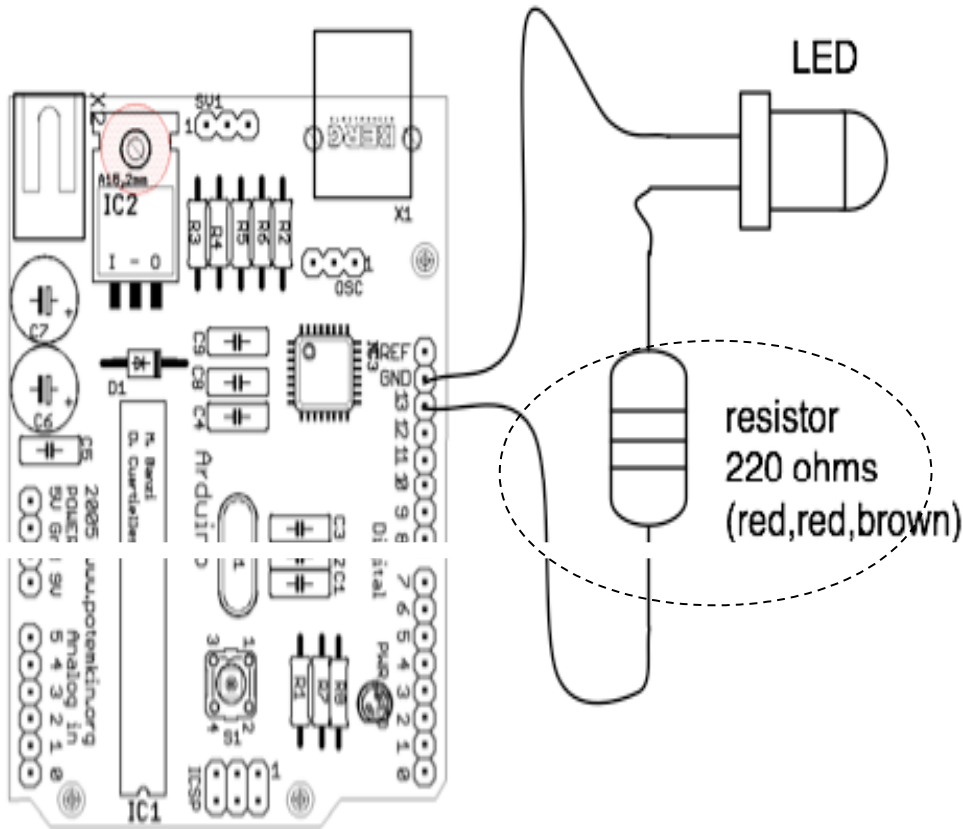
```
int switchPin = 2;      //(in photo pin 4) // digital input pin for a switch
int yellowLedPin = 3;  //(in photo pin 5) // digital output pin for a yellow LED
int redLedPin = 4;     //(in photo pin 6) // digital output pin for a red LED
boolean switchClose = false; // the state of the switch
```

void setup()

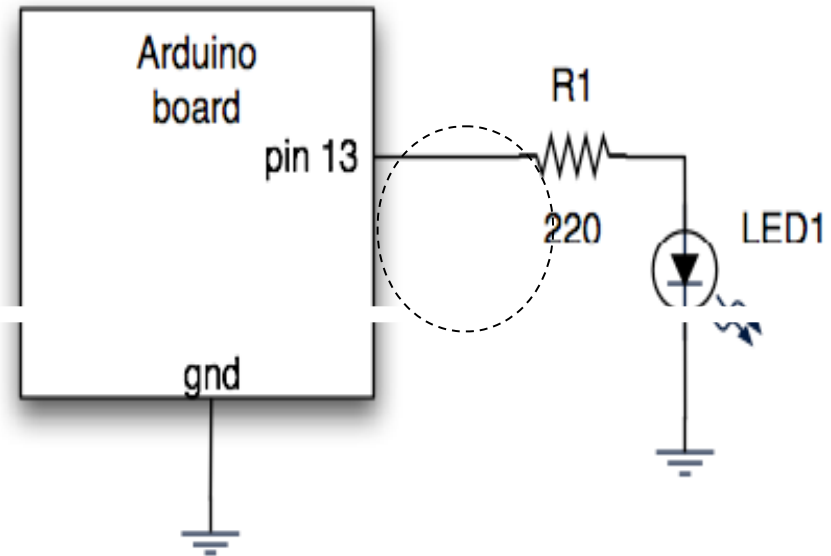
```
{  pinMode(switchPin, INPUT);    // set the switch pin to be an input
   pinMode(yellowLedPin, OUTPUT); // set the yellow LED pin to be an output
   pinMode(redLedPin, OUTPUT);   // set the red LED pin to be an output
} //setup
```

void loop()

```
{  switchClose = digitalRead(switchPin); // read the switch input:
   if (switchClose )                    // if the switch is closed:
   {  digitalWrite(yellowLedPin, HIGH); // turn on the yellow LED
      digitalWrite(redLedPin, LOW);     // turn off the red LED
   }
   else // if the switch is open:
   {  digitalWrite(yellowLedPin, LOW); // turn off the yellow LED
      digitalWrite(redLedPin, HIGH);  // turn on the red LED
   }
} //loop
```



wiring diagram



schematic

Arduino Sketch Structure

- ◆ Declare variables at top
- ◆ Initialize
 - ◆ `setup()` - run once at beginning, set pins
- ◆ Running
 - ◆ `loop()` - run repeatedly, after `setup()`

An Arduino “Sketch”

Global Variables

★ Declare variables at top

setup()

★ Initialize

☆ setup() – run once at beginning, set pins

loop()

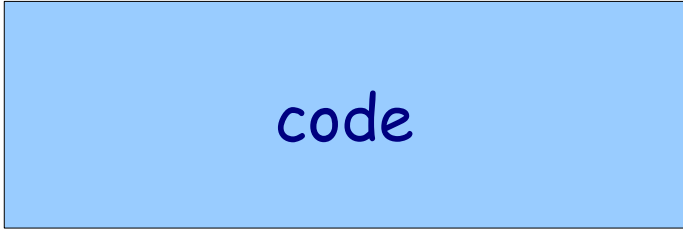
★ Running

☆ loop() – run repeatedly, after setup()

The program

```
void setup()
```

```
{
```



code

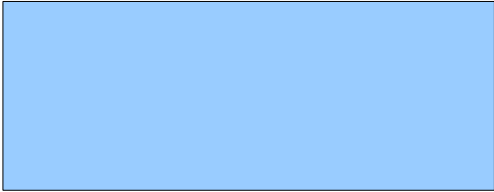
```
}
```

The setup() function is called when your program starts.

Use it to initialize your variables, pin modes, start using libraries, etc.

```
void setup() {                                //Inizio della configurazione
    pinMode(13,OUTPUT) ; //PIN 13 come output
}                                                //Termine della configurazione
```

The program

```
void loop()  
{  
    
}
```

The loop() function loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

The program

```
void setup() //Inizio della configurazione
{
    pinMode(13,OUTPUT); //PIN 13 come output
} //Termine della configurazione

void loop() //Ciclo principale
{
    digitalWrite(13, HIGH); // Emette 5V sul pin 13
    Delay(200);              // Aspetta 200 millisecondi
    digitalWrite(13,LOW);    // Emette 0V sul PIN 13
    delay(100);              // Aspetta 100 millisecondi
} //Termine del ciclo principale
```

Reference library

`digitalWrite(pin, value)`

Description

- Outputs either HIGH or LOW at a specified pin.

Parameters

- pin: the pin number
- value: HIGH or LOW

Returns

- none

```
digitalWrite(13, HIGH);    //Emette 5V sul pin 13
```

Defining Pin Levels, HIGH and LOW

When reading or writing to a digital pin there are only two possible values a pin can take/be-set-to: HIGH and LOW.

- **HIGH** represents the programming equivalent to 5 volts. When reading the value at a digital pin if there is 3 volts or more at the input pin, the microprocessor will understand it as HIGH. This constant is also represented by the integer number 1, and also the truth level TRUE.
- **LOW** is representing the programming equivalent to 0 volts. When reading the value at a digital pin, if we get 2 volts or less, the microprocessor will understand it as LOW. This constant is also represented by the integer number 0, and also the truth level FALSE.

Reference library

Delay(ms)

Description

- Pauses your program for the amount of time (in milliseconds) specified as parameter.

Parameters

- ms: the number of milliseconds to pause (there are 1000 milliseconds in a second)

Returns

- nothing

```
delay(200) ;    //Aspetta 200 millisecondi
```

The language

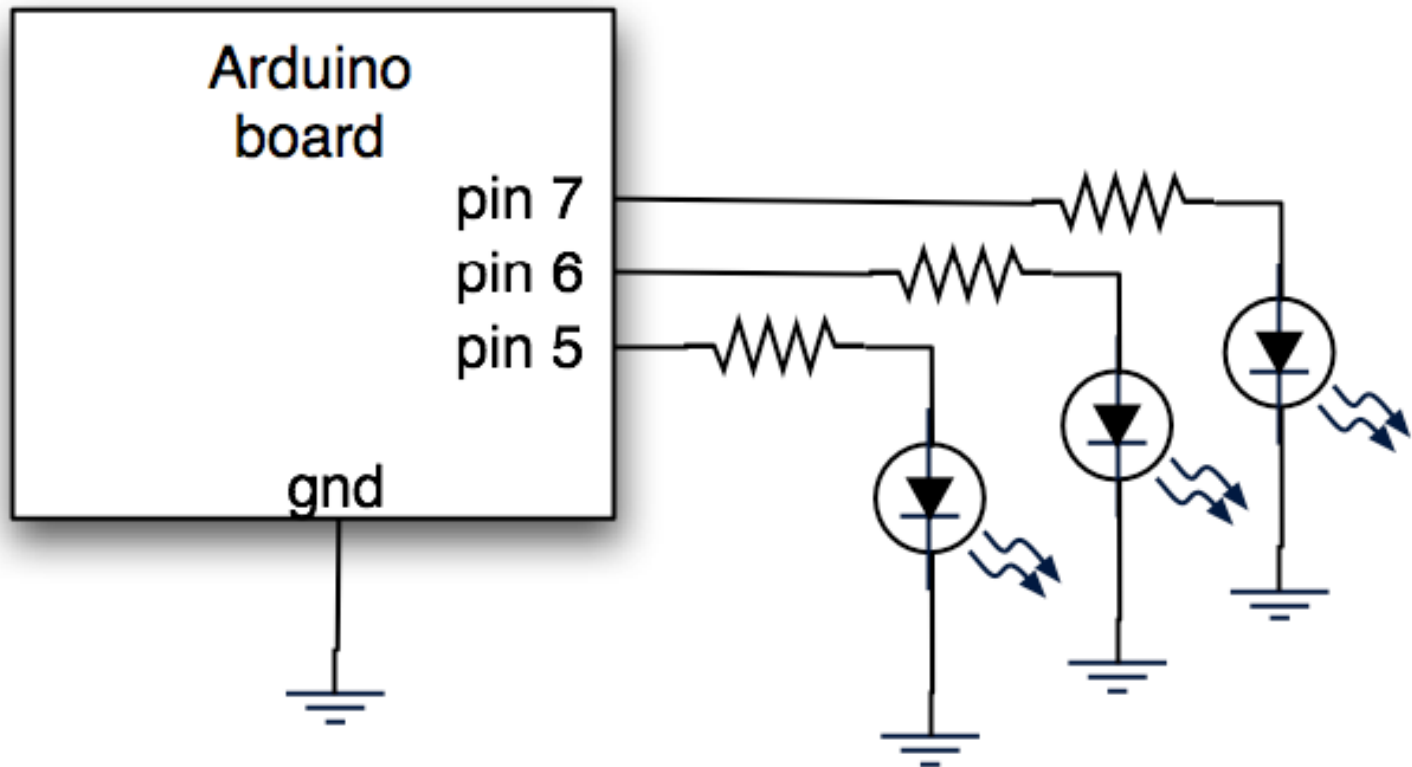
Language is standard C and C++ (but made easy)

Lots of useful functions:

- ◆ **PinMode(...)** - set a pin as input or output
- ◆ **DigitalWrite(...)** - set a digital pin high/low
- ◆ **DigitalRead(...)** - read a digital pin's state
- ◆ **AnalogRead(...)** - read an analog pin
- ◆ **AnalogWrite(...)** - write an "analog" PWM value
- ◆ **Delay(...)** - wait an amount of time
- ◆ **millis()** - get the current time

And many others. And libraries. And examples!

Exercise: More Blinky Madness



Exercise: SOS

A LED blinks, emitting a SOS in Morse code:
three dits, three dahs, and three dits

(· · · - - - · · ·) "3 dots" "3 dashes" "3 dots"

In popular usage, SOS became associated with
such phrases as "save our ship", "save our
souls" and "send out succour"

You need:

- A LED and a resistance of 220 ohms.
- A piezoelectric buzzer for a variant of the exercise, with audio effects.

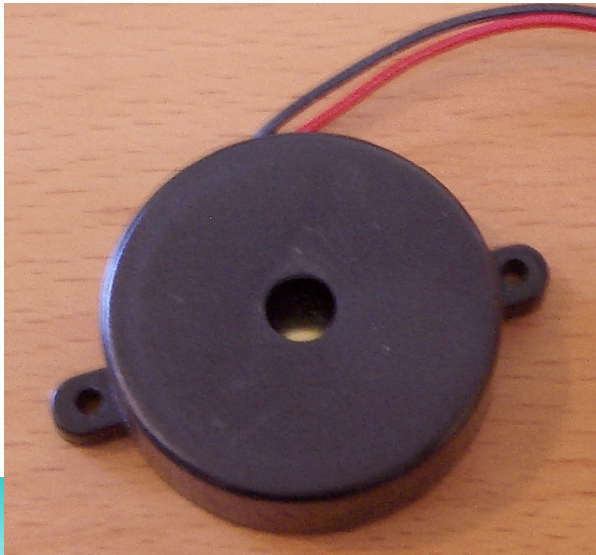
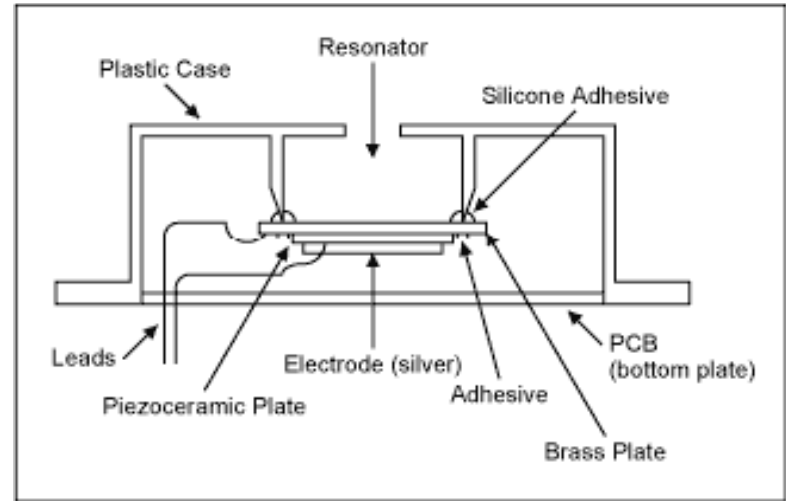
A piezoelectric sounder (buzzer)

A **buzzer** or **beeper** is a signaling device, typically a ceramic-based piezoelectric sounder like a Sonalert which makes a high-pitched tone. Usually these were hooked up to "driver" circuits which varied the pitch of the sound or pulsed the sound on and off.

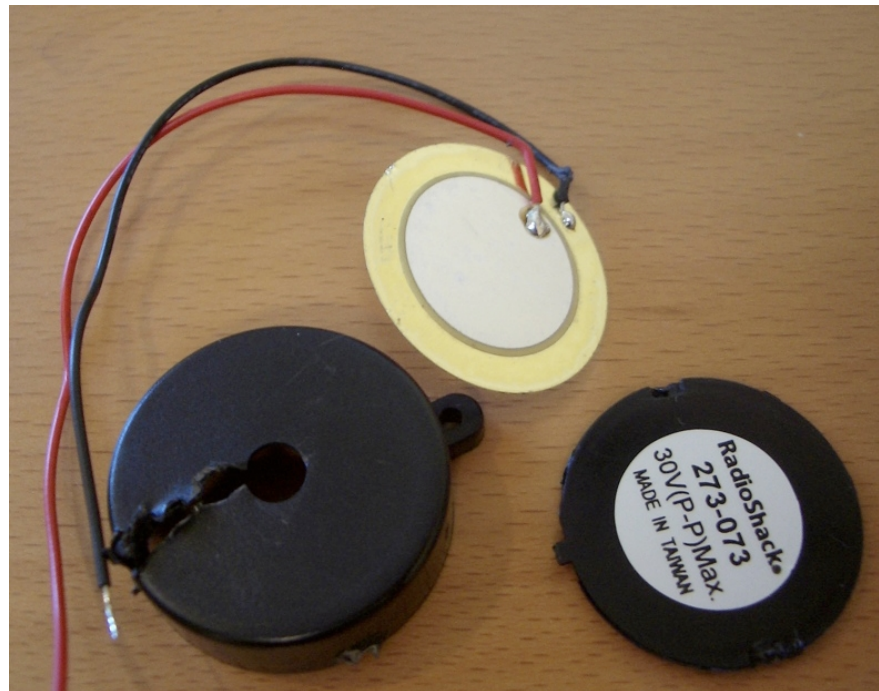
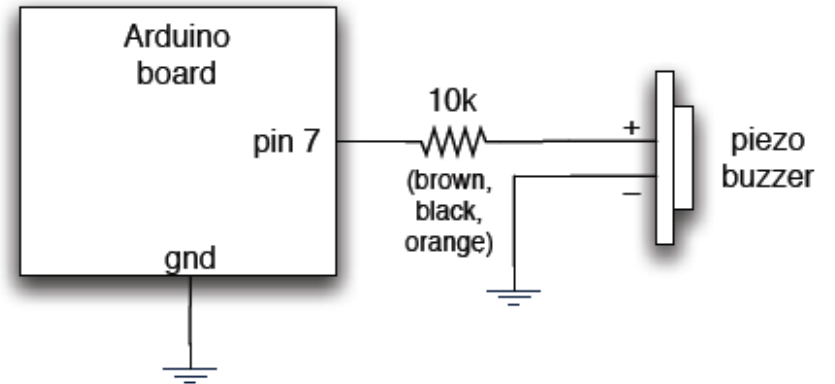
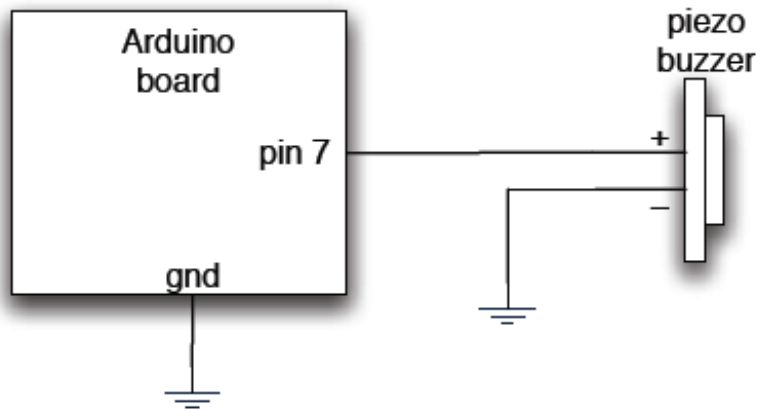


Piezoelectrics

- Some crystals, when squeezed, make a spark
- Piezo buzzers use this to make sound (flex something back and forth, it moves air)



- Two wires, red & black
- Apply an oscillating voltage to make a noise
- The buzzer case supports the piezo element and has resonant cavity for sound



```
const int unitLength = 200; // 200 msec: slow operator
const int dotLength = unitLength; //
const int dashLength = 3* unitLength ; //
const int partSpace= unitLength; //
const int lettersSpace= 3 * unitLength; //
const int wordSpace= 7 * unitLength; //
```

```
const int ledPin = 13;
const int buzzerPin = 10;
```

```
void setup ( )
{
    pinMode( ledPin ,OUTPUT ); //
    pinMode( buzzerPin ,OUTPUT ); //
}
```

```
/**
```

```
@brief definizione di procedura per generare il segnale S ( . . . )  
*/
```

```
void threeDots( )
```

```
{   for(int n=0; n <3; n++)
```

```
    { digitalWrite(ledPin, HIGH);
```

```
      digitalWrite(buzzerPin, HIGH);
```

```
      delay( dotLength );
```

```
      digitalWrite(ledPin,LOW);
```

```
      digitalWrite(buzzerPin,LOW);
```

```
      delay( partSpace );
```

```
    }
```

```
delay( lettersSpace - partSpace );
```

```
}
```

/**

@brief definizione di procedura per generare il segnale O (_ _ _)

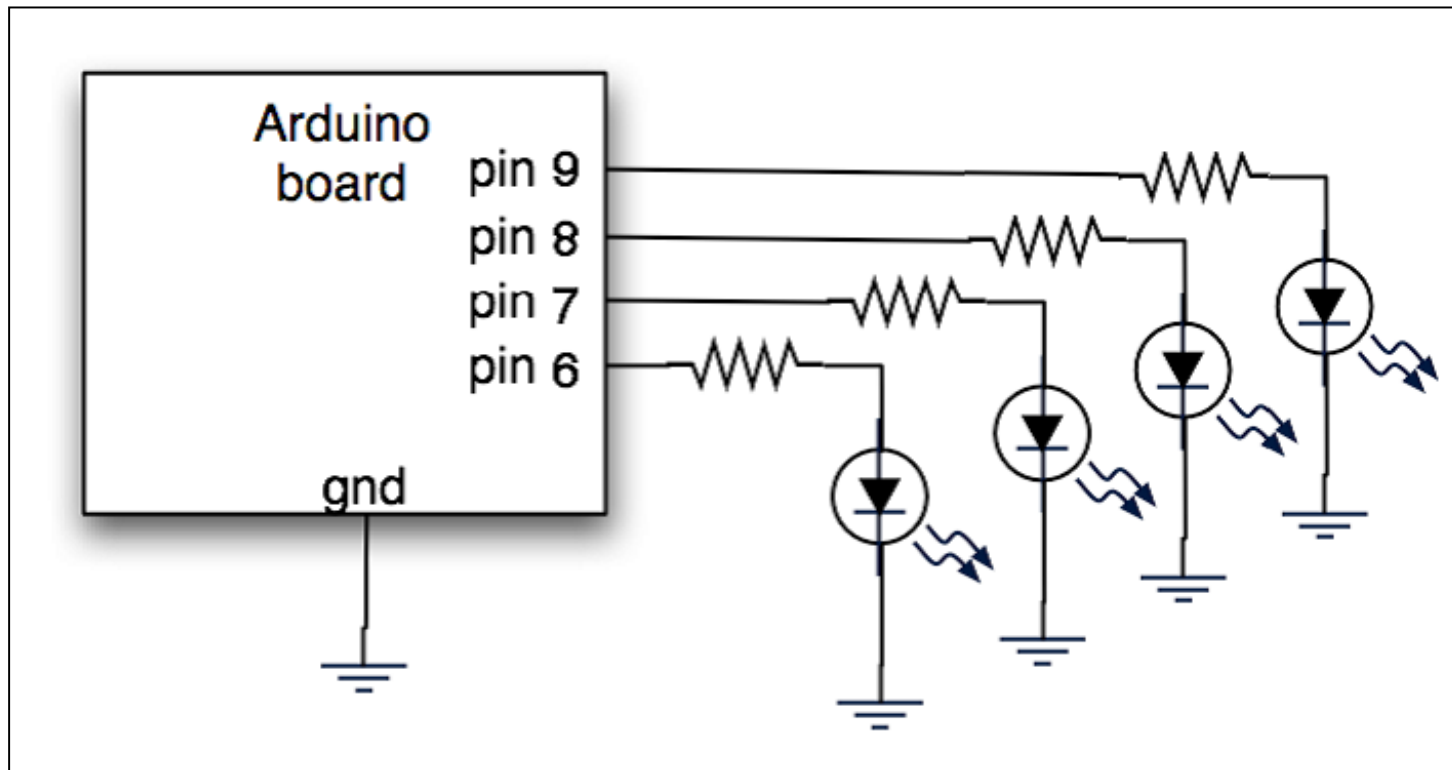
*/

```
void threeDashes( )
{   for ( int n= 0; n<3; n++ )
    {   digitalWrite( ledPin, HIGH );
        digitalWrite( buzzerPin, HIGH );
        delay( dashLength );
        digitalWrite( ledPin, LOW );
        digitalWrite( buzzerPin, LOW );
        delay(partSpace);
    } //for
    delay( lettersSpace - partSpace );
}
```

```
void loop( )  
{  
  threeDots( ); // S signal  
  
  threeDashes( ); // O signal  
  
  threeDots( ); // S signal  
  
  delay( wordSpace );  
}
```

Exercise

The 4 LEDs blink in unison. You need: four resistance of 220 ohms .



// declare variables:

```
int catenaLed[ ] = { 6, 7, 8, 9 };    // definition of the Led pins array  
int tempo = 500;
```

```
void setup()
```

```
{   for (int i=0; i<4; i++)           // all the pins as OUTPUT  
    { pinMode( catenaLed[ i ],OUTPUT); }  
}
```

```
void flash( )
```

```
{   for (int i=0; i<4; i++)  
    digitalWrite(catenaLed[ i ], HIGH ); // turn on the LED i  
  
    delay(tempo);  
  
    for (int i=0; i<4; i++)  
    digitalWrite( catenaLed[ i ], LOW );    // turn off the LED i  
  
    delay(tempo);  
}
```

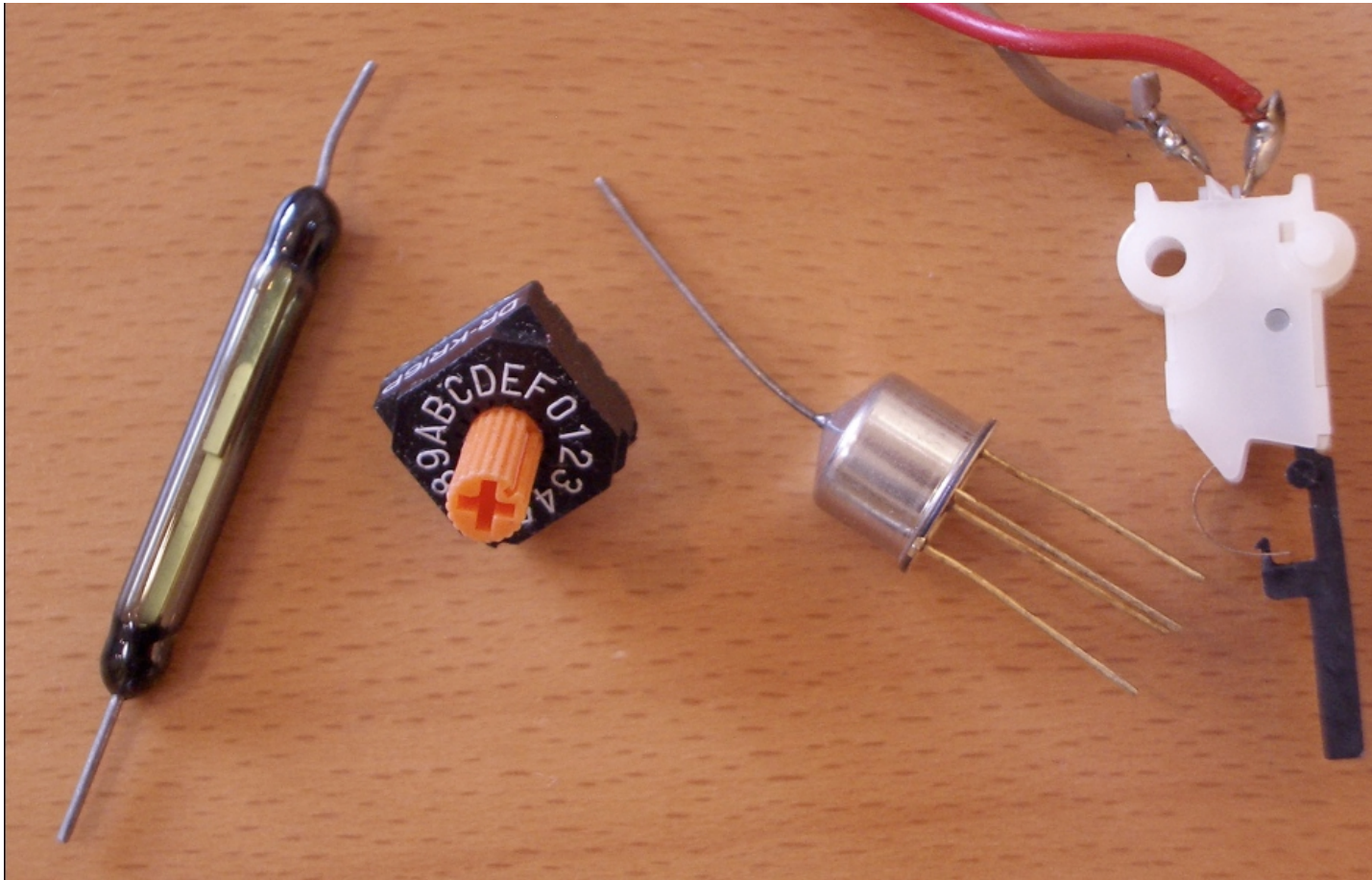
```
void loop( ) { flash( ); }
```

Digital Input

- ◆ Most inputs you'll use are variations on switches
 - ◆ Switches make or break a connection
 - ◆ Single pole = only one circuit is being controlled
-
- Double pole = two circuits are being controlled at once
 - Single throw = only one path for circuit
 - Double throw = two potential paths for circuit



Many Kinds of Switches

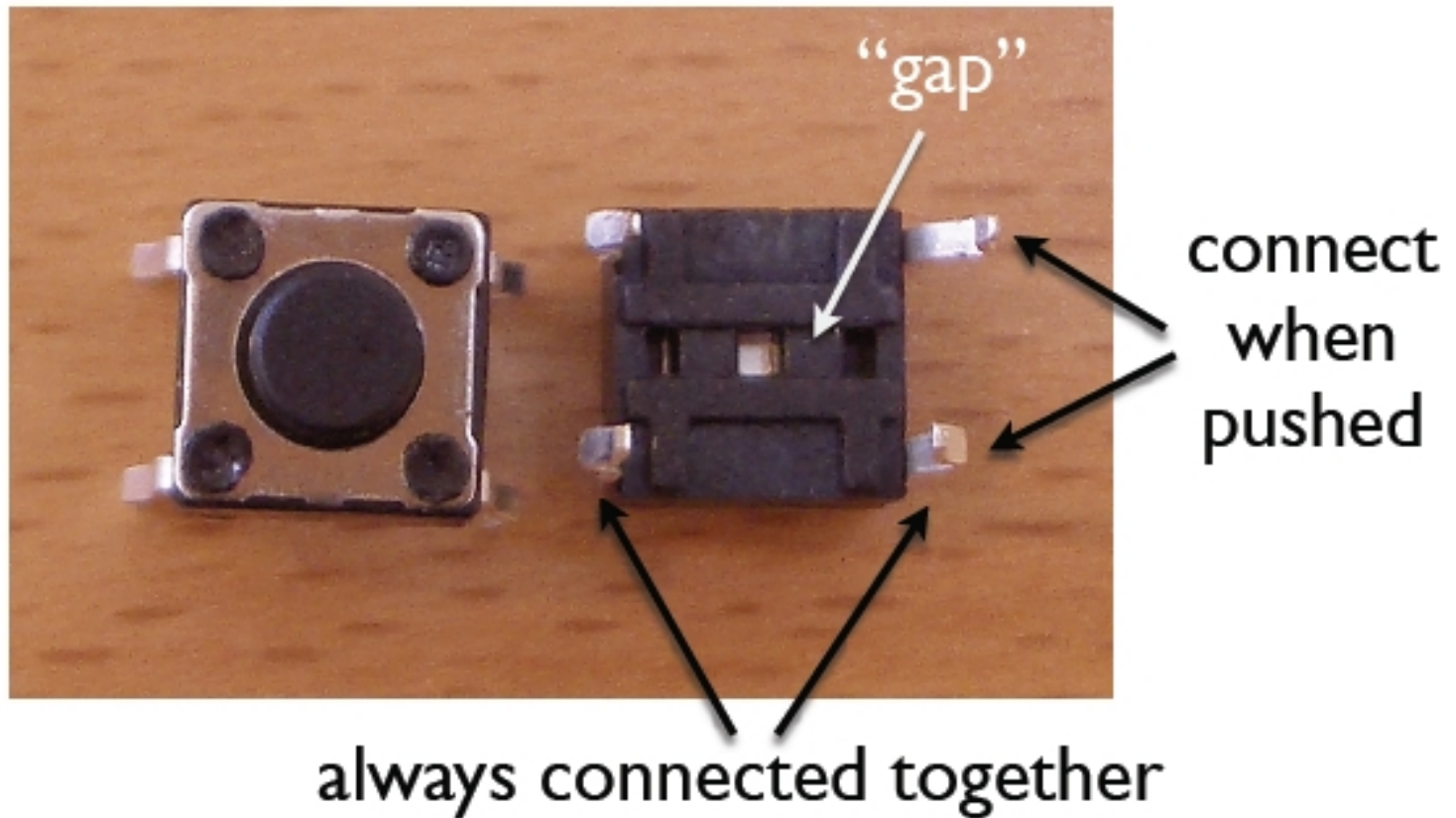


magnetic hexadecimal

tilt

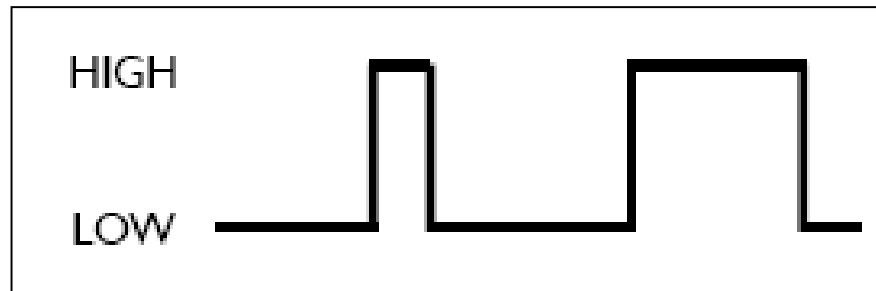
lever

Tiny Switches



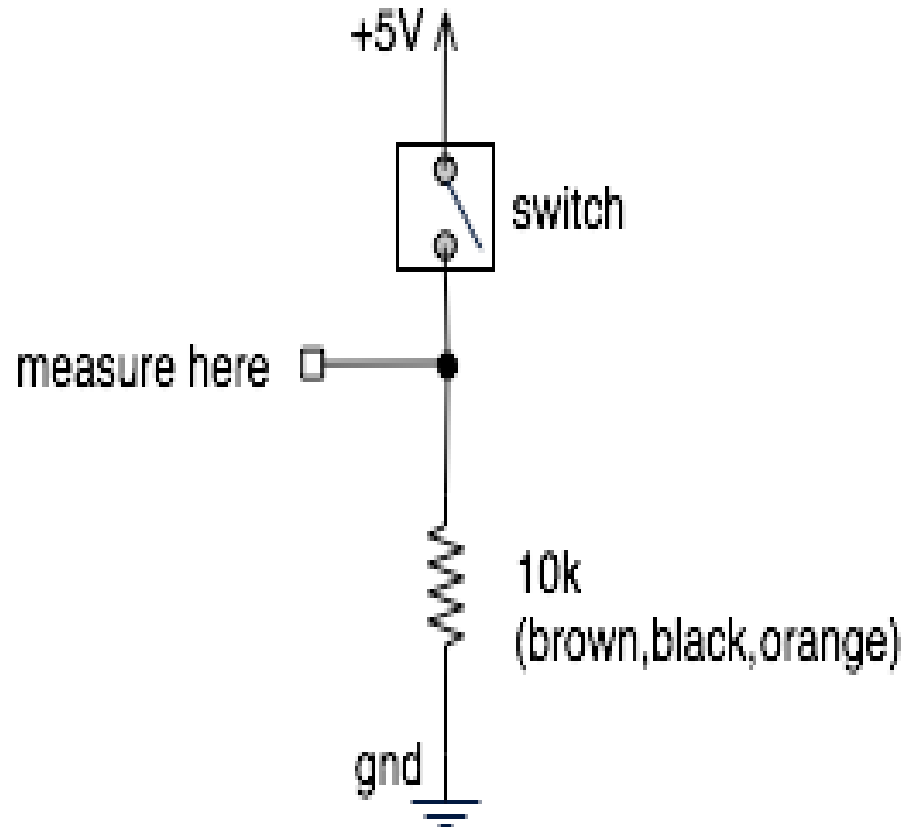
Digital Input

- ◆ Switches make or break a connection
- ◆ But Arduino wants to see a voltage
 - ◆ Specifically, a "**HIGH**" (5 volts or 3.3 volts)
 - ◆ or a "**LOW**" (0 volts)



Switch to Volts: Positive Logic

- Digital inputs can “float” between 0 and 5 volts
- Resistor “pulls down” input to ground (0 volts)
- Pressing switch sets input to 5 volts
- **Press is HIGH**
Release is LOW



Example from Arduino IDE: File > Examples>Communications>ASCII table

/*

ASCII table

Prints out byte values in all possible formats:

- * as raw binary values
- * as ASCII-encoded decimal, hex, octal, and binary values

For more on ASCII, see <http://www.asciitable.com> and
<http://en.wikipedia.org/wiki/ASCII>

The circuit: No external hardware needed.

created 2006
by Nicholas Zambetti
modified 9 Apr 2012
by Tom Igoe

This example code is in the public domain.

<<http://www.zambetti.com>>

*/

```
void setup()
{ //Initialize serial and wait for port to open:
  Serial.begin(9600);
  while (! Serial)
  { ; // wait for serial port to connect. Needed for Leonardo only
  }

  // prints title with ending line break
  Serial.println("ASCII Table ~ Character Map");
}

// first visible ASCIIcharacter '!' is number 33:

int thisByte = 33;

// you can also write ASCII characters in single quotes.
// for example. '!' is the same as 33, so you could also use this:
// int thisByte = '!';
```

```
void loop()
```

```
{
```

```
  // prints value unaltered, i.e. the raw binary version of the  
  // byte. The serial monitor interprets all bytes as  
  // ASCII, so 33, the first number, will show up as '!'
```

```
  Serial.write(thisByte);
```

```
  Serial.print(", dec: ");
```

```
  // prints value as string as an ASCII-encoded decimal (base 10).  
  // Decimal is the default format for Serial.print()  
  // and Serial.println(), so no modifier is needed:
```

```
  Serial.print(thisByte);
```

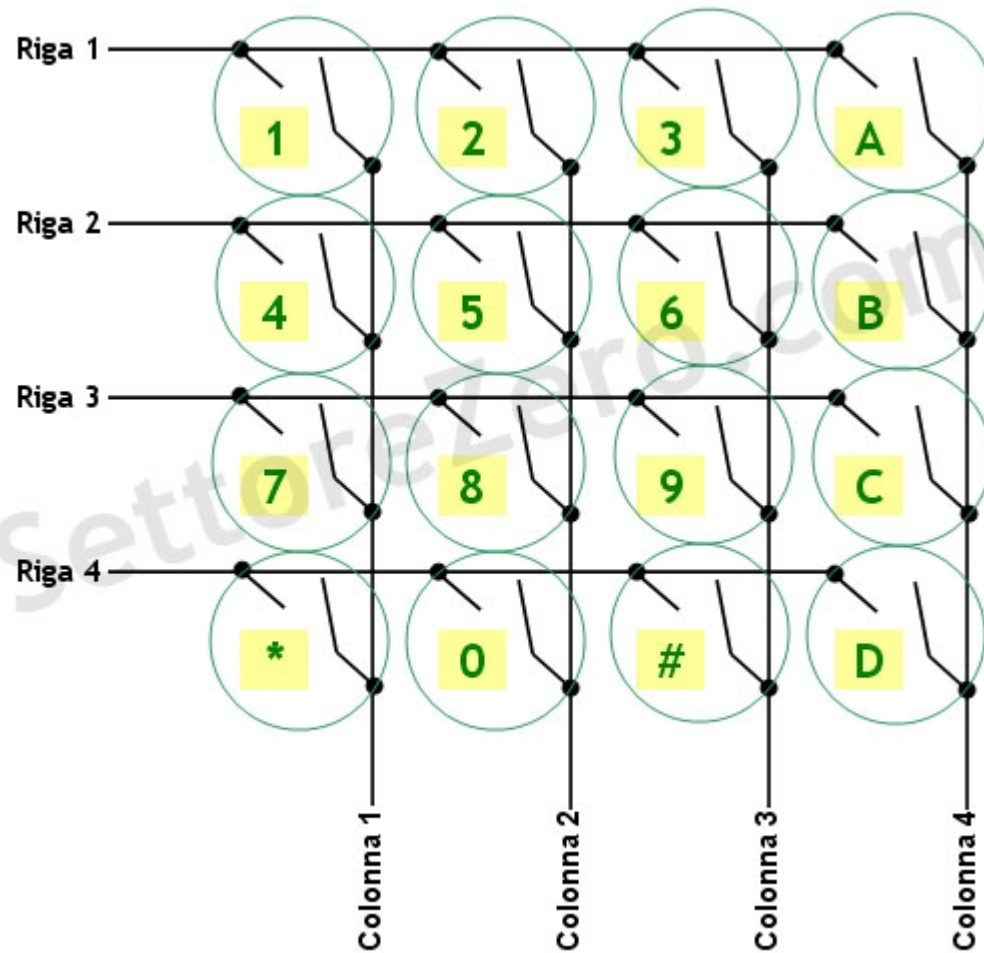
```
  // But you can declare the modifier for decimal if you want to.  
  // this also works if you uncomment it:  
  // Serial.print(thisByte, DEC);
```

```
Serial.print(" , hex: ");  
// prints value as string in hexadecimal (base 16):  
Serial.print(thisByte, HEX);
```

```
Serial.print(" , oct: ");  
// prints value as string in octal (base 8):  
Serial.print(thisByte, OCT);
```

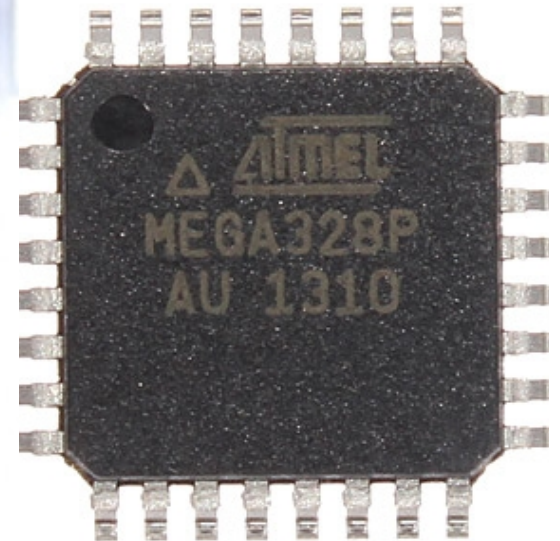
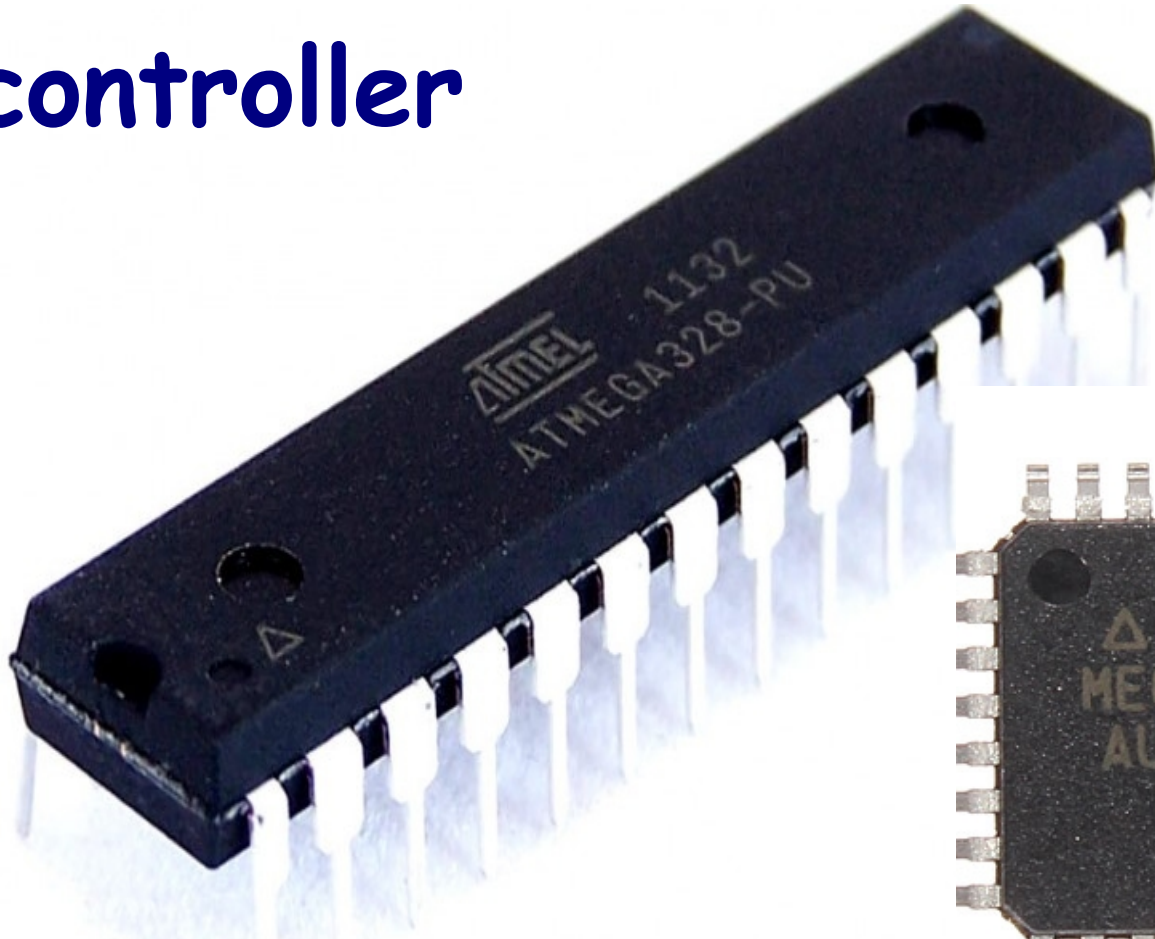
```
Serial.print(" , bin: ");  
// prints value as string in binary (base 2)  
// also prints ending line break:  
Serial.println(thisByte, BIN);
```

```
// if printed last visible character '~' or 126, stop:  
if ( thisByte == 126) { // you could also use if (thisByte == '~')  
{ // This loop loops forever and does nothing  
  while( true ) { continue; }  
}  
// go on to the next character  
thisByte++;  
}
```



Part B

inside microcontroller



... continua
nelle prossime serie di slide ...

Part A

General

<http://www.arduino.cc>

Part B

Hardware (microcontroller)

<http://www.atmel.com/devices/ATMEGA328.aspx?tab=documents>

- ATMEL DATASHEET: ATmega48A/PA/88A/PA/168A/PA/328/P

http://en.wikipedia.org/wiki/Atmel_AVR

<http://courses.cs.washington.edu/courses/csep567/10wi/lectures/>

Part C

ASSEMBLY

http://www.avr-asm-tutorial.net/avr_en/AVR_TUT.html

<http://www.avrbeginners.net/>

<http://www.cs.nmsu.edu/~jcook/arduino/index.php?n=Notes.AssemblyMods>

http://www.nongnu.org/avr-libc/user-manual/inline_asm.html